

AD-A055 387

INCO INC MCLEAN VA  
TERMINAL ACCESS PROCEDURES (TAP).(U)  
APR 78 K WELLS, M HUBERT, N HAUSER

F/G 9/2

UNCLASSIFIED

INCO/1090-178-FR-3-D(F)

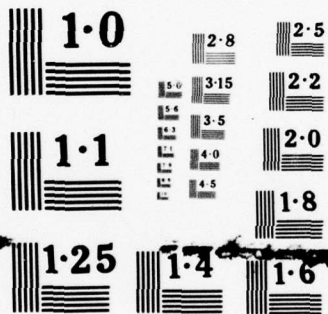
RADC-TR-78-90

F30602-77-C-0045

NL

1 OF 2  
ADA  
055387





NATIONAL BUREAU OF STANDARDS  
MICROCOPY RESOLUTION TEST CHART



FOR FURTHER TRAN

AD A 055387

RADC-TR-78-90  
Final Technical Report  
April 1978

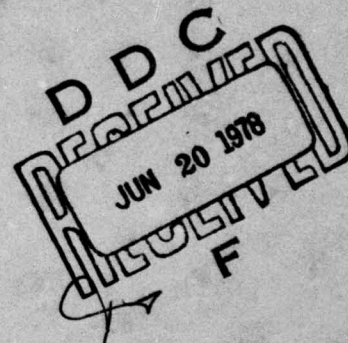
(2)



TERMINAL ACCESS PROCEDURES (TAP)

Kenyon Wells  
Margery Hubert  
Neal Hauser

INCO Incorporated



Approved for public release; distribution unlimited.

AD No. \_\_\_\_\_  
DDC FILE COPY

ROME AIR DEVELOPMENT CENTER  
Air Force Systems Command  
Griffiss Air Force Base, New York 13441

78 06 07 013

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-78-90 has been reviewed and is approved for publication.

APPROVED:

*Patricia M Langendorf*

PATRICIA LANGENDORF  
Project Engineer

APPROVED:

*Howard Davis*

HOWARD DAVIS  
Technical Director  
Intelligence & Reconnaissance Division

FOR THE COMMANDER:

*John P. Huss*

JOHN P. HUSS  
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRDA) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-78-90	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) TERMINAL ACCESS PROCEDURES (TAP)	5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 15 Jan 77 - 14 Jan 78	6. PERFORMING ORG. REPORT NUMBER INCO/1090-178-FR-3-D(F)
7. AUTHOR(s) Kenyon Wells, Margery Hubert Neal Hauser	8. CONTRACT OR GRANT NUMBER(s) F30602-77-C-0045 new	9. PERFORMING ORGANIZATION NAME AND ADDRESS INCO, Incorporated 7916 Westpark Drive McLean VA 22101
10. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRDA) Griffiss AFB NY 13441	11. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 45941024	12. REPORT DATE April 1978
13. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same	14. NUMBER OF PAGES 118	15. SECURITY CLASS. (of this report) UNCLASSIFIED
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.	17. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Patricia Langendorf (IRDA)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Data Dictionary Distributed Data Base Query Language Transparency		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This Final Technical Report details the design and development of the Terminal Access Procedures (TAP) of the Transparent Integrated Intelligence Network (TIIN). The TAP project is an integral part of the long-range development of TIIN, a highly user-oriented distributed data access and processing capability encompassing world-wide resources.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

407 275

act

# TABLE OF CONTENTS

		<u>Page No.</u>
SECTION I	INTRODUCTION	I-1
	1. Overview	I-1
	2. Development Approach	I-1
	3. Project Objectives	I-3
	4. Research and Development Activities	I-3
	5. Interactive Network Inquiry Processor	I-4
	6. Report Organization	I-5
SECTION II	NETWORK ACCESS DIRECTORY	II-1
	1. Introduction	II-1
	2. Composition of the NAD	II-2
	3. Functions of the NAD	II-8
	4. Future Development	II-12
SECTION III	NAD MAINTENANCE AND INTERROGATION	III-1
	1. Introduction	III-1
	2. Analyst AIDs Processor	III-1
	3. User Data Description Processor	III-5
	4. Host Data Description Processor	III-20
	5. Future Development	III-26
SECTION IV	QUERY LANGUAGE PROCESSOR	IV-1
	1. Introduction	IV-1
	2. Transparency Examples Language	IV-1
	3. Data Request Intermediate Format	IV-7
	4. QLP Capabilities	IV-18
	5. QLP Design	IV-20
APPENDIX A	DEVELOPMENT BACKGROUND	A-1
APPENDIX B	DESCRIPTION OF THE TIIN SYSTEM	B-1
APPENDIX C	GLOSSARY	C-1
APPENDIX D	IMPLEMENTATION LANGUAGE	D-1
APPENDIX E	DBMS INTERFACE	E-1
APPENDIX F	BIBLIOGRAPHY	F-1

ACCESSION for	
NTIS	Write Section <input checked="" type="checkbox"/>
DDC	Built Section <input type="checkbox"/>
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
CIAL	

78

iii

06 07 013

# LIST OF ILLUSTRATIONS

<u>Figure No.</u>	<u>Title</u>	<u>Page No.</u>
I-1	Transparent Integrated Intelligence Network Modules	I-2
III-1	Analyst Aids Processor Informational Levels	III-3
III-2	Analyst Aids Processor-Hierarchical Module Relationship with Displays	III-6
III-3	Structure of Data Definition at User NAD	III-8
IV-1	Syntax Description of TAP Common Query Language	IV-2
IV-2	Layout of DRIF and QNF	IV-10
IV-3	Syntax Description of the Reverse Polish Notation (RPN) Used in the Conditional Table of the QNF	IV-12
IV-4	Standard Operator Codes in DRIF	IV-14
IV-5	Keyword Table	IV-23
IV-6	Operator Stack	IV-25
IV-7	Operand Stack	IV-26



The Terminal Access Procedures (TAP) subsystem is one of five subsystems encompassing the Transparent Integrated Intelligence Network (TIIN). The goal of the TIIN is to develop a system capable of providing to the intelligence analyst transparent access to a number of diverse data bases. The goal of the TAP subsystem is to develop the interface from TIIN users to the system. TIIN users fall into two groups: Analysts and data base administrators. Analysts are those using the TIIN as an information source whereas data base administrators are responsible for maintaining the currency and integrity of the TIIN data sources and data sinks. The TIIN/TAP processors developed for use by the analyst include the Analyst Aids Processor and the Query Language Processor. The Analyst Aids Processor is used in the process of query refinement. From broad based subject categories the analyst is able to pinpoint specific element names to be used in the query. The Query Language Processor is then used to submit the query. The TIIN/TAP processors developed for use by the data base administrator include the User Data Description Processor and the Host Data Description Processor. The User Data Description Processor is used to perform maintenance on the User NAD. Similarly, the Host Data Description Processor is used to perform maintenance on the Host NAD.

The first task undertaken was an evaluation, and where necessary modification was made of the Network Access Directory (NAD). The NAD is the repository of information required by the TIIN modules concerning location, content, and structure of network data bases. Individual versions of the NAD are maintained at each host and user node. A single Host NAD defines that portion of a particular host data base which is to be made part of the network data base. A user NAD defines that portion of the network data base which is accessible from the user node. This portion may vary from user node to user node depending on the requirements of users at each node as well as the processing capabilities available at each node. A detailed description of the structure and content of both the Host NAD and the User NAD is contained in Section II of the Final Technical Report.

After evaluating and modifying both the Host and User NAD, processors were developed to construct and maintain them. The Host Data Description Processor is used by the host data base administrator to build the Host NAD. The User Data Description Processor is used by the user data base administrator to build the User NAD. Each of these processors runs in the interactive mode with their respective data base administrators. Both processors are password protected to avoid alteration of any NAD by anyone other than an authorized data base administrator. Both processors also seek to maintain the integrity of the NAD by prohibiting the data base administrator from performing certain functions until all prerequisite functions are performed.

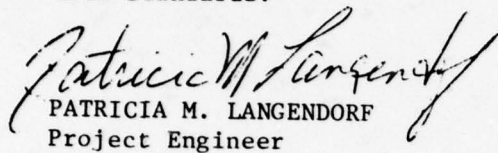
The Analyst Aids Processor was developed to enable the TIIN analyst to see descriptions of network data base contents. These descriptions are contained in each User NAD. They are used by the analyst to formulate his ideas into specific queries. At the highest level he is able to display

lists of general subject information from which specific subjects of interest are chosen. For each selected subject a list of files relevant to that subject may be displayed. From this list the analyst may select particular files of interest. For each of these files a description of the file is generated. The option also exists to display all of the file's member elements. The analyst is now ready to formulate a query or a series of queries.

The format for submitting a query is the Transparency Examples Language (TEL). It has been developed as an example of a standard query language. No knowledge of the location of data within the network is needed to submit a query. the TEL query language is designed around the Data Request Intermediate Form (DRIF) developed during the TIIN/QIP effort. DRIF is the internal query representation. Several nice features have been incorporated into the query language. One of these features is the incorporation of both a host selection phrase and a file selection phrase into the language. The host selection phrase is used to select a particular host data base towards which the query will be directed. The file selection phrase selects a particular data file. Both of these phrases are optional and if omitted the selection of a target data base and file will automatically be made by TIIN. Another feature is the capability to use either standard element names or locally defined user element names in the query. This allows a familiar set of element names to be used at each TIIN site. A third capability is provided by the responses phrase. It enables the analyst to limit the amount of output returned in response to his query. The Query Language Processor is extremely modularized allowing many other features to be added to the language as the need arises.

## EVALUATION

This report covers the terminal access subsystem necessary to provide an intelligence analyst transparent network access to diverse data bases. The primary contribution it makes to Air Force capabilities will probably be aid in identifying actual costs due to lack of network data standards.

  
PATRICIA M. LANGENDORF  
Project Engineer



## SECTION I

### INTRODUCTION

#### 1. OVERVIEW

This Final Technical Report details the design and development of the Terminal Access Procedures (TAP) of the Transparent Integrated Intelligence Network (TIIN) under Rome Air Development Center contract number F30602-77-C-0045. The project is an integral part of the long-range development of a Transparent Integrated Intelligence Network (TIIN), a highly user-oriented distributed data access and processing capability encompassing world-wide resources. The Final Technical Report is the culmination of the present TAP effort. More detailed information is found in the TAP System/Subsystem Specifications and the TAP Computer Program Documentation. An overview of the TIIN system and its development background is found in Appendices A and B of this report. Figure I-1 gives an overview of TIIN.

#### 2. DEVELOPMENT APPROACH

The TIIN/TAP Project is INCO's fourth contract effort concerned with the development of the Transparent Integrated Intelligence Network. During the first phase of the project, research focussed on two distinct areas. The first area was that of the requirements placed on the Network Access Directory (NAD) by the TIIN subsystems. The NAD should be designed to maximize its utility by each TIIN module. The second area involved in specification and development of a common query language. The first specification of this query language was done in the TAP Functional Description released in July 1977. Also included in this document was an initial list of functions to be performed by the Network Data Catalog Processor.

As the project progressed the list of specific requirements to be met by the NAD was built and a new NAD design proposed. The new design incorporates the originally separate Network Data Catalog into a unified, but expanded, NAD. The Network Data Catalog Processor was expanded in scope to include maintenance of both User and Host NADs as well as providing NAD information to the TIIN analyst. The processor was then divided into three processors, each related to a particular set of functions. The Analyst Aids Processor displays NAD information useful in query development to the TIIN Analyst. The User Data Description Processor aids the User DBA in performing User NAD maintenance. The Host Data Description Processor aids the Host DBA in performing Host NAD maintenance. Each of these was documented in technical memoranda.

The next step involved design of each of the four processors. The design of each proceeded in a top-down structured fashion.

Finally the processors were coded using RATFOR as the implementation language and TTDL as the terminal interface language. The writing of this Final Technical Report, the System/Subsystem Specifications, and the Program Documentation concludes this project.

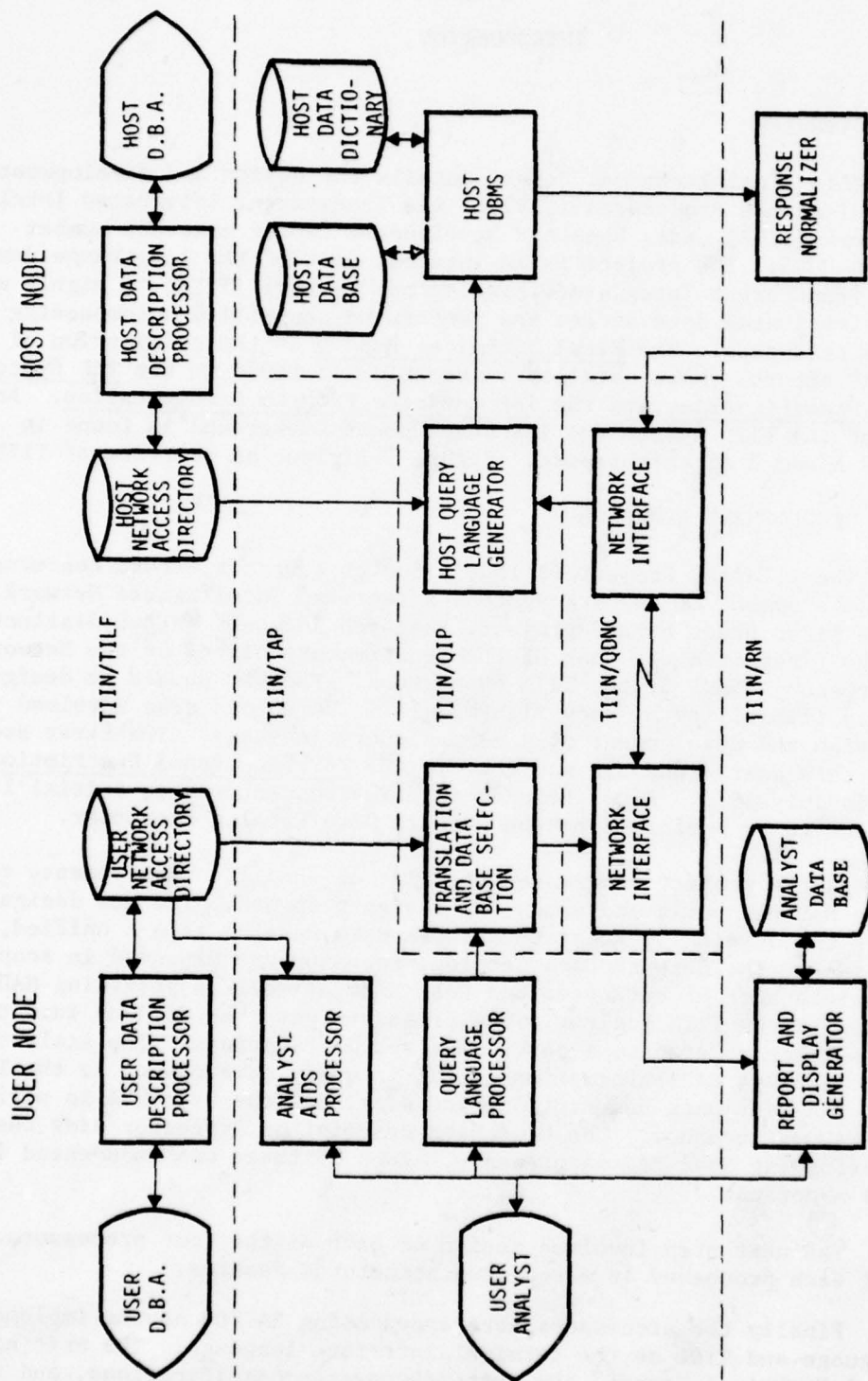


Figure I-1. Transparent Integrated Intelligence Network Modules

3. PROJECT OBJECTIVES

The tasks and objectives of the TIIN/TAP project were:

- o to analyze the NAD and determine its ability to meet the requirements of the TIIN subsystems,
- o to make any needed changes to the NAD based on the requirements as determined in the first task,
- o to design, develop and implement a processor which will allow an analyst to retrieve information useful in formulating a query from the NAD,
- o to design, develop and implement a processor to aid the DBA in maintaining the NAD,
- o to develop an example of a user query language,
- o to design, develop and implement a query language processor for translating standard query language into DRIF (Data Request Intermediate Form),
- o to investigate the requirements for an interactive Network Inquiry Processor which will guide the analyst through the process of making an inquiry without recourse to a formal language.

4. RESEARCH AND DEVELOPMENT ACTIVITIES

The following research and development activities were performed in compliance with the contract:

- o analysis of NAD content and structure,
- o development of distinct Host and User NAD to meet the requirements of the Host and User TIIN modules,
- o documentation of NAD design,
- o development of an Analyst Aids Processor to be used by the TIIN Analyst for displaying useful NAD information,
- o development of a Host Data Description Processor which allows the Host DBA to build and maintain the Host NAD,



- o development of a User Data Description Processor which allows the User DBA to build and maintain the User NAD
- o development of a BNF specification of the TIIN standard query language
- o development of a Query Language Processor for the TIIN standard query language
- o delivery of the Functional Description
- o delivery of the Test Plan and Procedures
- o delivery of the Final Technical Report
- o delivery of the System/Subsystem Specifications
- o delivery of the Program Documentation
- o delivery of the four processors coded in RATFOR.

#### 5. INTERACTIVE NETWORK INQUIRY PROCESSOR

The TAP system is one of the five subsystems comprising TIIN. Each of these five subsystems focuses on a particular portion of the TIIN design. The initial focus of the TAP effort was strictly towards providing methods for aiding the analyst in the query submission process. The three major tasks defined were:

- o The development and implementation of a Network Data Catalog and a Network Data Catalog Processor used by the analyst in the determination of particular data bases and particular files of interest. This determination could be made with only a limited prior knowledge of subject areas of interest.
- o The development and implementation of a Query Language Processor which will translate the Transparency Examples Language into DRIF (the internal query representation).
- o The development and implementation of an Interactive Network Inquiry Processor which would develop the DRIF query format from the analyst's responses to questions posed by the processor.

As the TAP project began, the project engineer pointed out that the third task was little more than a combination of the first two and provided no new required capability to the TIIN system. The designs with which the project should be concerned were those which provide the basic

TIIN capabilities such as a common query language, the ability to query several heterogeneous data bases, the ability to merge response files from multiple host data bases into a comprehensive report, etc. This direction required the planned TAP effort on the Interactive Network Inquiry Processor to be redirected towards the more important goals of the overall TIIN project. The following areas of development summarize this redirection.

- o An evaluation of each of the subsystem's requirements for the NAD was performed resulting in the separation of the NAD into Host and User components. The results of this effort are found in Section II of this report.
- o The highly redundant Network Data Catalog was eliminated by merging it with the User NAD. This decision and resultant design is documented in both Sections II and III of this report.
- o Two new processors, the User Data Description Processor and the Host Data Description Processor, have been added to the TIIN systems. These processors build and maintain the User NAD and Host NAD, respectively. A description of these processors is found in Sections III-3 and III-4.
- o A determination was made of future enhancements to both the Analyst Aids Processor and the Query Language Processor which would, to a large extent, provide the capabilities specified for the Interactive Network Inquiry Processor. These enhancements are documented in Sections III-5 and IV-4.
- o Specifications for a set of DBMS interface routines were developed. These routines will be used in the future design of all TIIN modules requiring file management services. All of the routines specified assume the hierarchical file structure described in the TIIN Response Normalization, Final Technical Report. A description of these routines is found in Appendix E of this report.
- o A High Order Language was selected for use as a design tool to be used in clearly defining the design of future TIIN modules. The language is described in Appendix D of this report.

## 6. REPORT ORGANIZATION

This report describes the TAP research and development work which was performed between January 1977, when the contract began, and January 1978 when it expired. The four sections of the report in conjunction with the appendices cover all phases of the research performed under the

TIIN/TAP contract. The reader who is unfamiliar with the TIIN system should read Appendices A, B and C to familiarize himself with the whole system before reading the rest of the report. The paragraphs which follow give a brief description of each section and each appendix of the report.

a. Section I: Introduction

The first section gives an overview of the goals and accomplishments of the TAP effort.

b. Section II: Network Access Directory

This section provides a detailed description of the NAD and its functions within the TIIN. The concept of separate Host and User NAD is emphasized.

c. Section III: NAD Maintenance and Interrogation

This section describes the function and design of each of the three processors acting as interfaces between three separate groups of users and the NAD. The intelligence analyst uses the Analyst Aids Processor to display NAD information useful in query development. The Host DBA uses the Host Data Description Processor to build and maintain the Host NAD. The User DBA uses the User Data Description Processor to build and maintain the User NAD.

d. Section IV: Query Language Translator

This section describes both the Transparency Examples Language and the translator which converts it to DRIF. A detailed discussion of the DRIF is also included.

e. Appendix A: Development Background

This appendix discusses the background and philosophy behind the development of TIIN.

f. Appendix B: Description of the TIIN System

This appendix discusses the functions provided by each of the modules in the TIIN system. Also included is a grouping of modules into their respective subsystems.

g. Appendix C: Glossary

This appendix is a glossary of terms relating to TIIN in particular and distributive processing in general.

h. Appendix D: Implementation Language

This appendix briefly discusses the features of the current TIIN implementation language.

i. Appendix E: DBMS Interface

This appendix describes the function of each of the subroutines used in TAP as interfaces to a DBMS.

j. Appendix F: Bibliography

This appendix lists published material which is relevant to TIIN/TAP.



## SECTION II

### NETWORK ACCESS DIRECTORY

#### 1. INTRODUCTION

The Network Access Directory (NAD) is the repository for all information required by the system about the contents of network data bases. This information will be used by TIIN in responding to an intelligence analyst's query. It will also be used by TIIN in response to the analyst's application programs which request data from the network. Included in this pool of information are the following items.

- o Data Location
- o Data File Contents
- o Standard Data Element Names
- o User Data Element Names
- o Host Data Element Names
- o Data Privacy Locks
- o Response File Structures
- o Value Conversion Algorithms
- o Data Files' Subject Content.

This information, as well as any additional information later deemed necessary, is contained in two functionally distinct portions of the NAD. The first portion has been named the User NAD. A single User NAD will be built for each user node in the network. It will contain the bulk of the data needed by all of the User TIIN modules to perform their respective functions. Information specifying the mapping from user element names to standard element names and information specifying the location of elements within the network is typically found here.

Informational content of the User NAD is controlled by the User DBA (or possibly several privileged users). In controlling the content of the NAD the User DBA has control over that portion of the network's data which is available to analysts at his site. If the User DBA does not include a data base, a data file, or a data element in the User NAD then that data is unavailable to TIIN analysts at the node.



The analyst who needs any of this unavailable data must know which host data base to query, which file to query and which host element names to use in his query. Without this exact knowledge the analyst is restricted to entering queries which the User DBA has chosen to allow.

The content of the Host NAD is controlled by the Host DBA (or possibly a set of privileged users). By exercising this control the Host DBA decides what portion of his Host Data Base will be made accessible to the network. He also determines the requirements, if any, an analyst must meet in order to access his portion of the network data.

The remainder of this section discusses in detail the content of each of the User and Host NAD files. The use of each of these files by the TIIN subsystems which have been investigated to date is also discussed. Finally, a discussion of future enhancements to the NAD and the reasons for each closes out this section.

## 2. COMPOSITON OF THE NAD

The NAD consists of a set of hierarchical files. Each Host NAD contains two of these files. The first file describes in detail the characteristics of each of the host elements. The second describes the response file format associated with each host data file. Each User NAD contains a minimum of five files. One of these files maintains information about each data base within the network to which the user has access. A group of files (one for each accessible host data base) contains information regarding the content of each host's data files. Two additional files maintain information about standard element names and values and user element names and values. Finally, a file used as a thesaurus of NAD subject information is maintained.

### a. User NAD

The User NAD is the primary source of information about data location and data format for all TIIN modules labeled user modules. It contains information about all data elements which are accessible to users at this node. If information concerning any particular data element is not contained in the User NAD for a node then that data is inaccessible to TIIN components at that node. This allows the scope of knowledge of users at any given node to be determined by the Data Base Administrator for that node. He is the individual who will decide what will and what will not be added to the User NAD. The number of data bases to which the User Data Base Administrator may grant access is limited only by the number of host data bases which have been made available to the TIIN network and the storage space available to the User NAD.

In the following descriptions of NAD files the name of the file is given first. This item is followed by the description of a record. The number to the left of each text description

represent the hierarchical level of the item given within the record. The underlined items are data element names. The non-underlined items are names of the segments whose elements immediately follow. At any given hierarchical level the data element names will precede the segment names for the following level. The \* to the left of the level indicator indicates that this particular segment may have more than one occurrence within the parent segment. The text in parenthesis to the right of a data element name indicates the type of the corresponding data element value. Unless otherwise specified the key to each record is the first level 1 element listed. The key to a level n segment is the first element at level n + 1.

(1) Data Base Location File

The Data Base Location File contains one record for each host data base known at this particular user node. Each record in this file contains three elements. The Host Data Base Identifier specifies the identity of the Host NAD which contains detailed information about the Host Data Base to which this record applies. The DBMS field identifies the Data Base Management System which maintains this particular Data Base. Finally, the Text Description field contains a verbal description of this particular Host Data Base. This field is not used internally by TIIN but exists solely for the purpose of supplying descriptive information about the Data Base to an Intelligence Analyst.

DATA LOCATION BASE FILE

0) DATA BASE INFORMATION

- 1) DATA BASE IDENTIFIER (VARYING LENGTH CHARACTER STRING)
- 1) DBMS (VARYING LENGTH CHARACTER STRING)
- 1) TEXT DESCRIPTION (VARYING LENGTH CHARACTER STRING)

(2) Data Base Content File

A User NAD will contain one or more files of this type. One of these files will be created for each data base queriable from the node. Each one of these files is given

a name equivalent to the Data Base Identifier Element in the corresponding record in the Data Base Location File. Each record defines the content of a single Host file. It consists of the file name, a list of file elements, a list of subjects related to the file and a text description of the file.

DATA BASE CONTENT FILE

0) FILE INFORMATION

1) FILE NAME (VARYING LENGTH CHARACTER STRING)

1) TEXT DESCRIPTION (VARYING LENGTH CHARACTER STRING)

\*1) ELEMENT INFORMATION

2) STANDARD ELEMENT NAME (VARYING LENGTH CHARACTER STRING)

\*1) SUBJECT INFORMATION

2) SUBJECT NAME (VARYING LENGTH CHARACTER STRING)

(3) Standard Element File

This file is used to define each TIIN standard element available at this node. If an analyst enters a query containing standard element names which are not contained in this file, the query will be rejected. It should be pointed out here that the fact that an element is present and associated with a file and a Data Base in a User NAD does not guarantee that it is available to the user. It must also be contained in the Host NAD which defines the Data Base in question and the user must meet the particular requirements for access which the Host Data Base Administrator ultimately determines. The fields contained in each record include the element name, type, unit of measurement and text description. The element type will specify whether the element is a character string, an integer, a date, a pair of coordinates, etc. The unit of measurement field specifies the standard element unit of measurement. Finally, associated with each element is a list of Data Base Identifier - File Name pairs which are used to determine in which files the element is contained.



#### STANDARD ELEMENT FILE

##### 0) STANDARD ELEMENT INFORMATION

- 1) STANDARD ELEMENT NAME (VARYING LENGTH CHARACTER STRING)
- 1) STANDARD ELEMENT TYPE (INTEGER)
- 1) STANDARD ELEMENT UNIT OF MEASUREMENT (INTEGER)
- 1) TEXT DESCRIPTION (VARYING LENGTH CHARACTER STRING)

##### \*1) FILE INFORMATION

- 2) DATA BASE IDENTIFIER (VARYING LENGTH CHARACTER STRING)
- 2) FILE NAME (VARYING LENGTH CHARACTER STRING)

#### (4) User Element File

This file defines all local User elements. It exists so that a User Data Base Administrator may define elements which are local to his node only. The User Elements file provides for the mapping of User Element Names to Standard Element Names and User Element Values to Standard Element Values. Each record contains the User Element Name, Type, and Unit of Measurement. It also contains the associated Standard Element Name and a Value Conversion Algorithm Identifier.

#### USER ELEMENT FILE

##### 0) USER ELEMENT INFORMATION

- 1) USER ELEMENT NAME (VARYING LENGTH CHARACTER STRING)
- 1) USER ELEMENT TYPE (INTEGER)
- 1) USER ELEMENT UNIT OF MEASUREMENT (INTEGER)
- 1) STANDARD ELEMENT NAME (VARYING LENGTH CHARACTER STRING)
- 1) VALUE CONVERSION ALGORITHM IDENTIFIER (INTEGER)

(5) Subject File

This file exists solely for the purpose of aiding an analyst in formulating his query. Internally this file is not used for query or response translation or routing. It is used to enable an analyst to display the files associated with his subject selection. A record in the file consists of the Subject Name and a list of Data Base Identifier - File Name pairs specifying all files which cover the subject.

SUBJECT FILE

Ø) SUBJECT INFORMATION

1) SUBJECT NAME (VARYING LENGTH CHARACTER STRING)

\*1) FILE INFORMATION

2) DATA BASE IDENTIFIER (VARYING LENGTH CHARACTER STRING)

2) FILE NAME (VARYING LENGTH CHARACTER STRING)

b. Host NAD

The Host NAD is the primary source of information for the TIIN modules responsible for standard-to-host query conversion and host-to-standard data stream conversion. The Host QIP will use the information in the Host NAD to convert standard element names into their equivalent host element names. Standard element values will be converted to Host element values via an algorithm chosen by a Host NAD entry. Only those elements whose names are in the NAD are legal entries in a query. By selectively adding element names to the Host NAD the Host Data Base Administrator decides which elements within the data base will be known to the network and, furthermore, which may be used for retrieval purposes only and which may be used for qualification purposes only. This gives the Host Data Base Administrator control over the privacy of his data. The description which follows of the various files comprising the host NAD uses the same documentation conventions as were used in the User NAD description.

(1) Host Data Base Description File

The Host Base Description File is used for translation of TIIN standard queries into host dependent queries. These queries

are then submitted to the host DBMS and a response is received. The Host Data Base Description File will be used by the Response Normalization modules to produce a standardized response file.

A record in the Host Data Base Description File represents a single host file. It contains the file name and a list of elements available within the file. Each element entry contains the host element name, type, unit of measurement, and maximum value length (the length in bytes of the longest element). Also included is the corresponding standard element name, standard-to-host and host-to-standard value conversion algorithm identifiers and an identifier of the response file segment in which the element is contained. Finally, a variable length field is included which will contain the special characteristics associated with this particular data base. The length of this field will vary from Host NAD to Host NAD depending upon the Host Data Base in question.

#### HOST DATA BASE DESCRIPTION FILE

##### 0) FILE INFORMATION

- 1) FILE NAME (VARYING LENGTH CHARACTER STRING)

##### \*1) ELEMENT INFORMATION

- 2) HOST ELEMENT NAME (VARYING LENGTH CHARACTER STRING)
- 2) HOST ELEMENT TYPE (INTEGER)
- 2) HOST ELEMENT UNIT OF MEASUREMENT (INTEGER)
- 2) HOST ELEMENT MAXIMUM VALUE LENGTH (INTEGER)
- 2) STANDARD ELEMENT NAME (VARYING LENGTH CHARACTER STRING)
- 2) STANDARD TO HOST VALUE CONVERSION ALGORITHM IDENTIFIER (INTEGER)
- 2) HOST TO STANDARD VALUE CONVERSION ALGORITHM IDENTIFIER (INTEGER)
- 2) SEGMENT DESCRIPTOR IDENTIFIER (INTEGER)
- 2) SPECIAL CHARACTERISTICS (VARYING LENGTH BYTE STRING)



## (2) RNF Description File

Each record in this file describes the format of a single hierarchical response file. Each host file described in the Host NAD has a particular response file representation and thus a corresponding record in the RNF Description File. A record consists of a set of hierarchically related segment descriptors which in turn consist of a variable number of element descriptors. The segment descriptors define the hierarchical structure of the data record. The element descriptors define the name, type, maximum length, and unit of measurement of each of the response file's data elements. All of this descriptive information will be appended onto the front of every response file so that the file will be completely defined and able to be manipulated by TIIN at any node through which it passes.

### RNF DESCRIPTION FILE

#### Ø) FILE INFORMATION

- 1) FILE NAME (VARYING LENGTH CHARACTER STRING)

#### \*1) SEGMENT INFORMATION

- 2) SEGMENT IDENTIFIER (INTEGER)
- 2) PARENT SEGMENT IDENTIFIER (INTEGER)

#### \*2) ELEMENT INFORMATION

- 3) STANDARD ELEMENT NAME (VARYING LENGTH CHARACTER STRING)
- 3) STANDARD ELEMENT TYPE (INTEGER)
- 3) STANDARD ELEMENT UNIT OF MEASUREMENT (INTEGER)
- 3) STANDARD ELEMENT MAXIMUM VALUE LENGTH (INTEGER)

## 3. FUNCTIONS OF THE NAD

The goal of the TIIN development effort is the design of a system which will allow a user transparent access to a multitude of data base management systems. This means that an analyst with minimal knowledge of

query languages, data base management systems, and communications protocols will be able to sit down at a TIIN terminal and get answers to his questions. The completeness of the answers will depend on the amount of information within the network which has been made available to the analyst. Factors constraining information retrieval include the following:

- o The analyst's personal security clearance is not at the proper level to view or query data
- o Access to the appropriate data base has not been granted to the user's site. This may be for any number of reasons including security or a lack of computing facilities (i.e., disk storage facilities)
- o The analyst's query is so broad based that a return of the huge response file to the analyst's site would be prohibitively expensive. Too many network resources would be required for too long a length of time
- o The analyst is unable to enter an intelligible query even with the help of the TAP Analyst Aid Processor
- o The data base containing necessary response data has not been brought on-line to the network. This must be done by the data base administrator at the data base site.

In order to realize these goals several problems must be solved. These problems have been divided into five areas of study which have evolved into the five TIIN subsystems. These five subsystems are TILF (Transparent Intelligence Language Facility), QIP (Query Intermediate Processor), RN (Response Normalization), TAP (Terminal Access Procedures), and QDNC (Query Distribution Executive/Network Communications Interface). To date development has been done on the first four subsystems. QDNC remains to be started. The NAD requirements documented here deal only with the first four subsystems.

a. QIP Requirements

(1) QNF Generator

The QNF Generator converts the DRIF (Data Request Intermediate Format) into the QNF (Query Normal Format). Logically this step involves converting the query with user defined element names and values into the same query with standard element names and values. Also, host file selection is done by this module. The NAD dependent functions performed



include user element name to standard element name conversion, user element value to standard element value conversion, standard element name and value validation, and host file selection. The first two functions listed require a NAD mapping to user element names to standard element names. There is also a requirement for value conversion algorithm selection. Standard element name and standard element value validation require a list of all TIIN standard element names and their corresponding value representations. Host file selection requires a mapping of TIIN element names to their associated files. All of these functions are performed by the user QIP and all necessary data should be contained in the user section of the NAD.

#### (2) QTF Processor

The QTF Processor is responsible for translating standard element names and standard element values into their host file equivalents. The NAD requirement is mapping of standard element names to host element names for each host file. There is also a requirement for the NAD to provide for value translation algorithm selection so that standard element values may be translated into their host equivalents. All of this information logically belongs in the host section of the NAD because it is host file dependent.

#### b. RN Requirements

The Host RN is required to take a host DBMS report and convert it into a normalized response file. This requires the conversion of host data into an equivalent standard representation and the restructuring of a host DBMS report into a standardized file structure. The value conversion requires a mapping of host element names to standard element names and an associated value translation algorithm identifier. In order to restructure the host report into a normalized response file, the structure of the file, including the relationship between elements within each record, must be known. This information as applies to each host file must be present in the host portion of the NAD.

#### c. TAP Requirements

The TAP Analyst Aids Processor (AAP) will provide the analyst with the capability to peruse the descriptions of data base contents which are available at his particular node. This information should aid him in formulating a query. The list below enumerates the functions performed by the AAP. The required content of the NAD to allow the AAP to perform these functions is included with each function.

- o List all subjects available in alphabetical order. Allow the analyst to specify the beginning and ending character strings to be used in bounding the search. This requires a NAD file containing all subjects available at the node which is sorted in alphabetic sequence.
- o Given a particular subject, list all files containing information pertaining to that subject. This requires that each subject record in the subject NAD file contain an index of host files to which this subject applies.
- o Given a particular file, list all elements contained within it. This requires that each record in the NAD's Data Base Content File contain an index of TIIN elements which are contained in the host file as well as a text description of the host file.
- o Given a particular file, list all subjects to which the file applies. This requires a cross index of subjects contained within each file.
- o List description and format of TIIN standard element name. This requires a file of standard element names where each record contains the element name, a textual description of the element and a description of the element value format.

All of these NAD requirements pertain to the user portion of the NAD.

#### d. TILF Requirements

TILF makes no demands on either the Host NAD or the User NAD. It does, however, have the responsibility of maintaining the NAD. As a result of the concept of separate Host and User NAD's, a separate processor has been developed to handle the maintenance functions on each. These processors have been named the Host Data Description Processor (HDDP) and the User Data Description Processor (UDDP). They maintain the Host and User NADs respectively. Although both processors are functionally part of the TILF subsystem, they have been designed under the TAP contract due to the renovation of the NAD. The design of these two modules is described in Chapter III of this report.

#### 4. FUTURE DEVELOPMENT

##### a. Multifile Queries

A primary objective of TIIN development is the multifile query capability. This capability should exist under both user control and system control. User control provides the analyst with the necessary tools for specifying several partial queries aimed at particular host files together with a set of operators to be used in merging the multiple responses into a single response file. Under system control TIIN would automatically perform these operations given only a single query with no target file. Additions to both Host and User NADs are anticipated before these capabilities may be realized.

###### (1) Host NAD

At the host end, peculiarities of the various Host Data Base Management Systems must be determined. For each Host DBMS the special characteristics of the system which affect query submission or response retrieval must be noted and the appropriate information added to the Host NAD (provision has been made for just this with the inclusion of a Special Characteristics Field in the Host Elements File). In addition to DBMS specific information, the Host NAD must include additional information which will allow response file merger.

###### (2) User NAD

The User NAD must contain all information needed in the decomposition of a query into partial queries aimed at various host files. The required information for this operation is yet to be determined.

##### b. Data Access Controls

Once a determination is made of the desirable level of access control to host data, the required data for enforcing this control must be added to both Host and User NADs.

##### c. Update Procedures

Since the NAD is itself a Distributed Data Base, a method of maintaining its integrity as well as its informational timeliness must be determined. It is extremely important that an update made to a local NAD be made available to all other affected NADs.



## SECTION III

### NAD MAINTENANCE AND INTEROGATION

#### 1. INTRODUCTION

This section describes the NAD interrogation and update functions available to three classes of TIIN users. The first class of users consists of intelligence analysts who will be submitting queries to the system. They are concerned with getting answers to their questions. They are given the ability to display those portions of the NAD which may aid them in the query submission process. These users are not given the ability to modify the NAD in any way. A second class of users consists of User Data Base Administrators. Each is charged with the responsibility of maintaining a single User NAD. This task requires the ability to display, add to, and delete from the NAD. A set of modules is provided which perform these functions for the DBA while requiring minimal knowledge of the NAD structure from him. The third class of users consists of the Host Data Base Administrators. These users maintain the Host NAD's just as the User DBA's maintain the User NAD's and thus are provided with a similar set of functions.

#### 2. ANALYST AIDS PROCESSOR

The Analyst Aids Processor displays NAD information to the intelligence analyst. Specifically, this processor allows an analyst to display:

- o Alphabetical lists of subjects available at the node
- o Lists of files pertaining to particular subjects
- o Lists of subjects covered by particular files
- o Lists of elements contained within particular files
- o Descriptive information about any data base
- o Descriptive information about any file
- o Descriptive information about any element.

This information will be used by the analyst in the identification of files and elements which he will use within his query. Section III.2.a identifies the functions available to the analyst and their usage. Section III.2.b details the overall design of the processor.

a. Analyst Functions

An analyst's interrogation session will proceed through a sequence of displays. The displays are arranged by levels of information where the highest or entry level contains lists of subjects while the lowest level contains detailed file element information. This structure is shown in Figure III-1. At each step in the diagram the analyst has the choice of whether to go up or down a single level but may not advance through more than one level per command. The intent of this process is to guide the analyst from thoughts of general subject areas to specific file and element names which may be used to submit a query. The following paragraphs serve to clarify the actions taken by the analyst and the system when proceeding through this process.

An intelligence analyst's session with the Analyst Aids Processor begins with the following selection menu.

SELECT TRANSACTION TYPE:

1. LIST ALL SUBJECTS IN CATALOG.
2. LIST ALL SUBJECTS IN CATALOG WHICH BEGIN WITH CHARACTER STRING \_\_\_\_\_.
3. LIST ALL SUBJECTS IN CATALOG BETWEEN CHARACTER STRINGS \_\_\_\_\_ AND \_\_\_\_\_.
4. TERMINATE SESSION

ENTER YOUR CHOICE \_\_\_\_\_.

The analyst will then make a selection. If he has not decided to select all subjects for display he will see a display requesting the input of either one or two character strings to be used for bounding the list of subjects displayed. Now the subject list is displayed followed by this message.

ENTER SUBJECT OF INTEREST. IF NONE, ENTER NONE: \_\_\_\_\_.

If the analyst replies with "NONE" the original menu will be flashed and he may again request a list of subjects. If the analyst instead chooses a subject, a list of all files dealing with that particular subject will be displayed to him. He now knows which files may be of interest and is flashed the following:

ENTER FILE OF INTEREST. IF NONE, ENTER NONE \_\_\_\_\_.

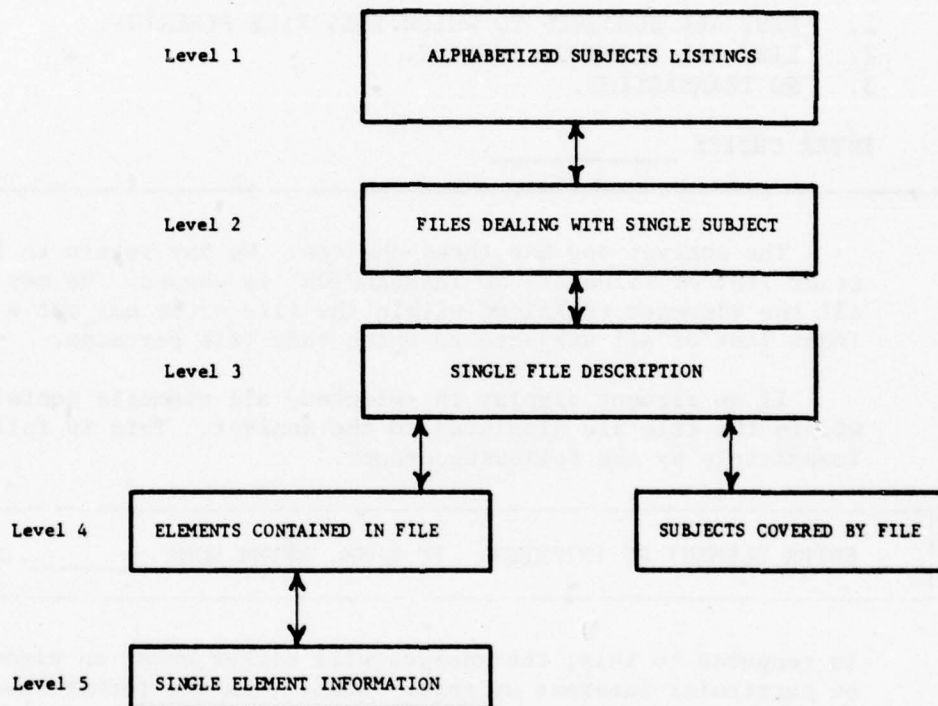


Figure III-1. Analyst Aids Processor Informational Levels

The analyst now has a list of all files pertaining to a particular subject in front of him and is asked to choose a particular file for more detailed information. If he is not interested in any one of the files, he enters "NONE" and the previous list of chosen subjects is again displayed. If a file selection is made, a description of that file is displayed in conjunction with the following selection menu.

SELECT TRANSACTION TYPE.

1. LIST ALL SUBJECTS TO WHICH THIS FILE PERTAINS.
2. LIST ALL ELEMENTS IN FILE.
3. NO TRANSACTION.

ENTER CHOICE \_\_\_\_\_.

The analyst now has three choices. He may return to the prior list of files if "NO TRANSACTION" is chosen. He may list all the elements contained within the file or he may get a cross index list of all subjects to which this file pertains.

If an element display is selected, all elements contained within the file are displayed to the analyst. This is followed immediately by the following prompt.

ENTER ELEMENT OF INTEREST. IF NONE, ENTER NONE\_\_\_\_\_.

In response to this, the analyst will either enter an element name of particular interest or enter "NONE." In the former case a detailed description of the element containing such items as element name, element type, unit of measurement, etc., is displayed to the analyst. This is followed by a redisplay of the element list and the prompt for choice of element. In the latter case, the user has entered "NONE," thus the system again displays the file information selection menu and again waits for the analyst's choice.

The analyst's final possible response to the file information selection menu is a request for a listing of all subjects to which the file pertains. The systems response to this request is a list of all subjects to which this file applies. At the end of this subject display the file information selection menu is again displayed and the analyst may make his next selection.



#### b. Processor Design

The Analyst Aids Processor is designed in a top-down fashion with each level of module nesting corresponding to an information level. The modules at any given level determine whether to call an additional module, thus increasing the nesting level or whether to return to the calling module, thus decreasing the nesting level based on the intelligence analyst's response to the current TTDL (Terminal Transparent Display Language) display. The calling of another routine is the response to an analyst's request for more specific data (i.e., from a file display, requesting a display of the elements in the file). A return to the invoking module is the response to an analyst's request for return to a prior, more generalized, display (i.e., from a display of specific information about an element in a file, return to the list of all elements in the file). Looping within a module is the response to an analyst's request to redefine the current display. This module nesting structure is shown in Figure III-2. The module name is listed together with the displays each module will flash to the user. A display marked as saved is unchanged in content until an analyst command forces a change. Thus, the same display may be flashed several times without requiring the analyst to define it each time. A one shot display is flashed once without its contents being preserved. In order to reflash the display the analyst must reenter the key information.

### 3. USER DATA DESCRIPTION PROCESSOR

The User Data Description Processor guides the User Data Base Administrator through the process of describing that portion of all network data bases which will be available at his node. This is accomplished during an interactive session. The processor prompts the DBA with a variety of menu displays and requests for input. The DBA's answers are used to build the User NAD. The User NAD then serves as a directory of network data which is now available at the User node.

#### a. User DBA Capabilities

##### (1) General Description

The User DBA has three major categories of information which he defines in the User NAD. These three categories are data description information, subject information, and user element information. The first category is by far the most significant. It includes the selection of the subset of network data bases, data files and data elements to be made available at the user node. This selection process proceeds in an interactive session. The DBA first defines each file's component elements. The rules which must be followed by the User DBA during this process are as follows:



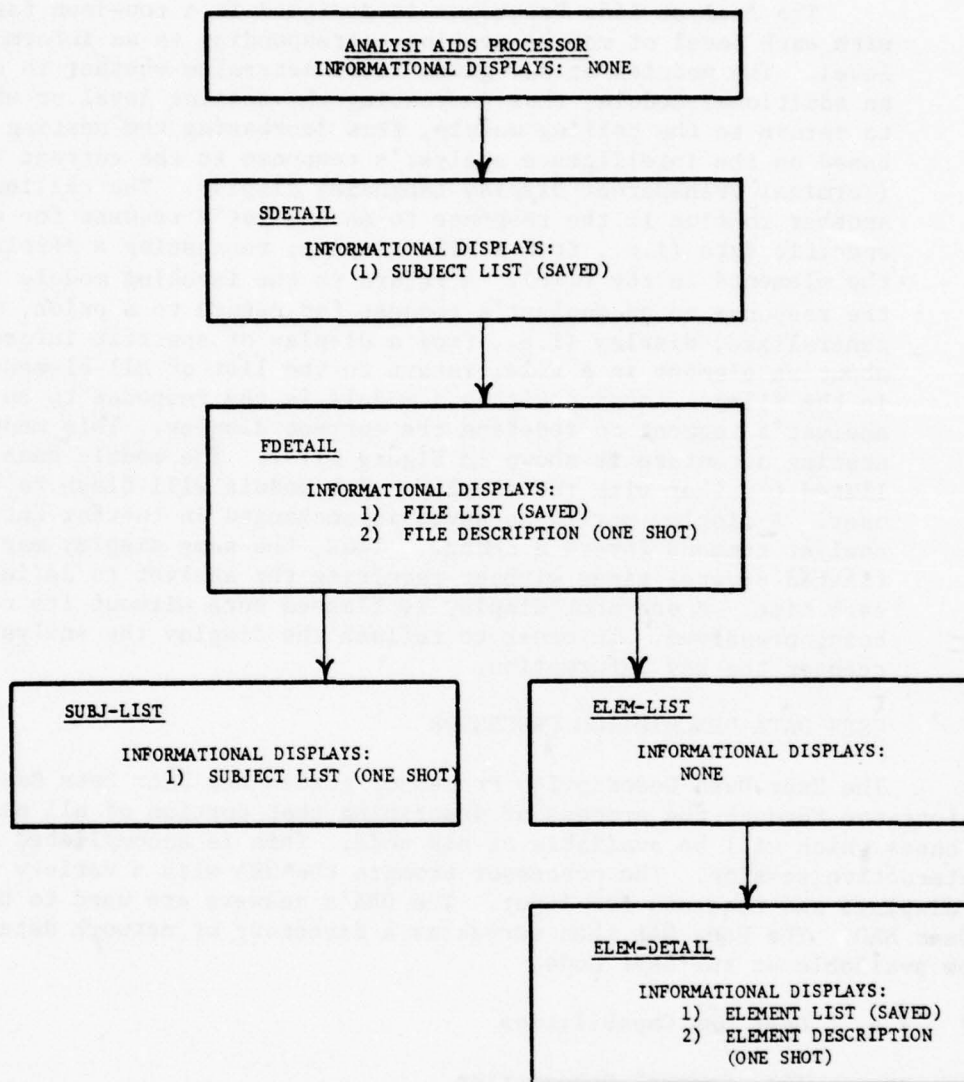


Figure III-2. Analyst Aids Processor - Hierarchical Module Relationships with Displays

- o A data base must be defined before any of its component files are defined
- o A file must be defined before any of its component elements are defined
- o An element must be defined in its standard representation before it is included as a file element.

These rules apply due to the relationships between data bases, files, and standard element names as shown in Figure III-3. The standard element names are the single global entity in the TIIN. Each file in the network will have its elements defined as standard elements at all user nodes having access to that file. Thus, the third rule requires the definition of the standard element before inclusion of it in any file. The first two rules are based on the structure of a data base. A data base consists of a set of data files each of which consists of a set of data elements. Each set is defined before its component elements are defined.

The second category of information maintained by the User DBA is subject information. It consists of an alphabetical list of topics, each cross referenced to a list of host data files. The User DBA will both define the list of subjects available at the User node and the mapping of these subjects to files which may be queried at the node. There is only one rule which the User DBA must follow when adding subject information to the NAD: The subject must be added to the NAD before it is tied to any data file. Thus, a User DBA will first define a new subject, which will cause it to be added to the alphabetized list of subjects, and then list the files to which the subject applies.

The third category of information maintained by the User DBA is user element information. This information makes it possible for an analyst or group of analysts to use their own element names and value representations in TIIN queries. They are not required to know the standard elements as long as the DBA has defined the needed user elements at the User node. In order to do this the DBA must know the standard element name to which the user element name maps and the identifier of the conversion algorithm which will be used to convert user element values into standard element values. The only rule affecting the addition of user elements to the NAD is that a standard element must be defined in the NAD before any user element which maps to it.

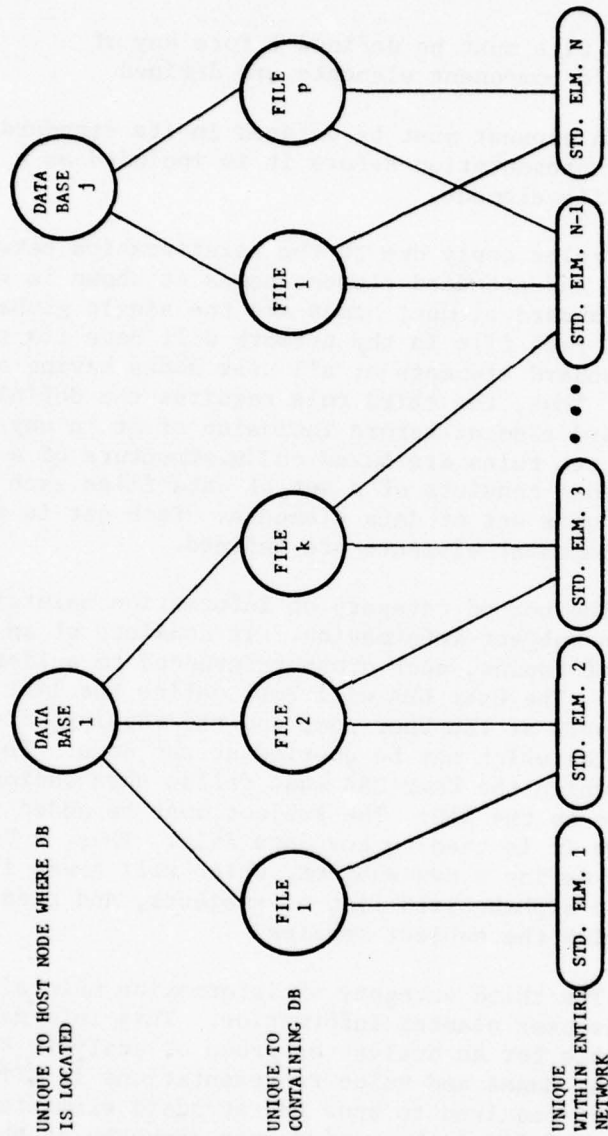


Figure III-3. Structure of Data Definition at User NAD.



## (2) Specific Capabilities

When the DBA activates the User Data Description Processor he will be flashed the following selection menu.

SELECT TRANSACTION TYPE	
1. DEFINE NAD	2. DEFINE DATA BASE
3. DEFINE FILE	4. DEFINE ELEMENT
5. DEFINE USER ELEMENT	6. DEFINE SUBJECT
7. DELETE NAD	8. DELETE DATA BASE
9. DELETE FILE	10. DELETE ELEMENT
11. DELETE USER ELEMENT	12. DELETE SUBJECT
13. ADD ELEMENT TO FILE	14. ADD SUBJECT TO FILE
15. DELETE ELEMENT FROM FILE	16. DELETE SUBJECT FROM FILE
17. DISPLAY INFORMATION	18. TERMINATE SESSION
ENTER YOUR CHOICE: _____	

The individual choices perform the following functions:

### 1: Define NAD

This function is used by the User DBA to create a User NAD. This function is normally only performed once per user node.

### 2: Define Data Base

This function is used to add a new data base to the list of those available for interrogation from the user node. Data bases not contained in the User NAD may only be queried by the use of a host file selection phrase and a host data base selection phrase in the query.

### 3: Define File

This function is used to add a new host file to the list of those available at the user node. The inclusion of a host file within a User NAD allows TIIN to query the file whenever the analyst enters a query pertinent to the file's contents.

### 4: Define Element

This function is used to add a standard element name to the NAD. The set of standard element names is used by the analysts in the development of queries.

5: Define User Element

This function is used to add User Elements to the NAD. User Elements are local to a user node and when used in a query will be translated into their standard element equivalents.

6: Define Subject

This function is used to add a new subject or topic of interest to the list within the User NAD. This subject list is used by the analyst in the determination of possible files of interest to him.

7: Delete NAD

This function will eliminate a User NAD from the TIIN system. Since all the information contained in the NAD is lost upon initiation of the function it should rarely be used.

8: Delete Data Base

This function causes a given data base and all of its files and elements to be deleted from the User NAD and thus deleted from the realm of data available to analysts at the node.

9: Delete File

This function causes a selected file and its elements to be deleted from a User NAD. The file is no longer available to a TIIN analyst at the node.

10: Delete Element

This function causes the deletion of a standard element from the list of elements contained within a particular file.

11: Delete User Element

This function deletes a user element from the list of those available at the user node. Once deleted the user element name may no longer be used in an analyst's query.

12: Delete Subject

This function deletes a subject from the list of those present in the NAD. A subject should be removed from each file to which it pertains before it is totally removed from the NAD.

#### 13: Add Element to File

This function adds a standard element to the list of elements contained within a particular file. An element must be defined as a standard element using command 4 before it is added to a file.

#### 14: Add Subject to File

This function adds a subject to the list of subjects covered by a particular file. This file will then be included in any analyst's display of files pertaining to the subject.

#### 15: Delete Element From File

This function deletes an element from the list of elements contained in a particular file. Once an element has been deleted the file is no longer interrogated by TIIN in response to a query containing that element.

#### 16: Delete Subject From File

This function will cause the deletion of a subject from any specified file. The specified file will no longer be displayed in the list of files pertaining to the subject.

#### 17: Display Information

This function will cause a second menu to be displayed to the DBA. This menu gives a choice of various User NAD information which may be displayed.

#### 18: Terminate Session

The terminate session function is used to terminate the User Data Description Processor.

#### (3) Individual Function Inputs

Most of the functions on the transaction selection menu will cause a second display to the analyst requesting further input. These displays and their expected inputs are defined below.

(a) The Define NAD function does not cause any display to be flashed to the analyst. It needs no input from the DBA because it creates a set of files but does not install any data in them.



(b) Define Data Base

This function causes the following to be displayed to the DBA.

ENTER THE FOLLOWING INFORMATION ABOUT THE DATA BASE

1. DATA BASE NAME \_\_\_\_\_
2. DATA BASE MANAGEMENT SYSTEM \_\_\_\_\_
3. TEXT DESCRIPTION OF DATA BASE \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

The DBA must then respond with the name of the data base (must uniquely identify it with network), the name of the data base management system maintaining the data base, and a verbal description of the data base. Once this information has been entered, the User Data Description Processor will add a record to the User NAD Data Bases File describing the data base. It will also create a file whose records will describe each of the new data base files. The file will be given the name of the new data base.

(c) Define File

This function causes the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE FILE.

1. NAME OF PARENT DATA BASE \_\_\_\_\_
2. NAME OF FILE \_\_\_\_\_
3. FILE DESCRIPTION \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

The parent data base must have been previously defined. It is the name of the data base containing the file. The file description is a text field to be displayed to any analyst requesting a description of the file. This function causes the addition of a record to the File File corresponding to the parent data base.

(d) Define Element

This function causes the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE ELEMENT.	
1. NAME OF ELEMENT	_____
2. ELEMENT TYPE	_____
3. ELEMENT UNIT OF MEASUREMENT	_____
4. ELEMENT DESCRIPTION	_____
	_____
	_____

The Name of Element field is a character string element name. The element type field contains an integer which represents the type of the element (types such as alphanumeric, real, numeric integer, boolean, etc.). The element unit of measurement field contains an integer which represents the unit of measurement (kilometers, feet, miles, pounds, etc.). The element description field contains a text description of the element which may be displayed to the intelligence analysts on request. The define element function causes a record to be added to the User NAD Standard Element File.

(e) Define User Element

This function causes the following display.

ENTER THE FOLLOWING INFORMATION TO DEFINE THE USER ELEMENT	
1. USER ELEMENT NAME	_____
2. USER ELEMENT TYPE	_____
3. USER ELEMENT UNIT OF MEASUREMENT	_____
4. VALUE CONVERSION ALGORITHM IDENTIFIER	_____
5. CORRESPONDING STANDARD ELEMENT NAME	_____

The first 3 input fields have identical input formats as the first 3 entries of the define element display. The value conversion algorithm identifier field contains an integer identifying the conversion algorithm used to convert the user element value to a standard element

value. The corresponding standard element name field contains the name of the standard element to which this user element maps. This information will be used to add a record to the User NAD User Element File.

(f) Define Subject

This function causes the following display.

ENTER SUBJECT NAME: \_\_\_\_\_

The subject input by the DBA is used to add a record to the subject file.

(g) Delete NAD

This function requires no input. It will destroy all of the files comprising the User NAD.

(h) Delete Data Base

This function will cause the following display.

ENTER THE DATA BASE NAME: \_\_\_\_\_

The data base name is used as the key in the deletion of the record in the Data Base File. Also the entire File File describing all files in the data base is deleted.

(i) Delete File

This function will cause the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE FILE

1. FILE NAME \_\_\_\_\_
2. PARENT DATA BASE NAME \_\_\_\_\_



The file name and parent data base name are both character strings. The parent data base name is used to locate the File File which describes all files contained in the data base. The file name is used to delete the record describing the chosen file from the File File. All standard Element file cross references and all subject file cross references are also deleted.

(j) Delete Element

This function causes the following display.

ENTER NAME OF ELEMENT: \_\_\_\_\_

The element name will be used to delete the record from the Standard Elements file. A check will be made to insure the element is not still contained in any host file. If any such file is found the DBA is warned and the element is not deleted.

(k) Delete User Element

This function causes the following display.

ENTER USER ELEMENT NAME: \_\_\_\_\_

The user element name is used to select the appropriate record from the User Element file for deletion. The record is then deleted from the file. No other file is affected.

(l) Delete Subject

This function causes the following display.

ENTER SUBJECT NAME: \_\_\_\_\_

The function will delete the specified subject from the subject file. If the subject is still cross referenced to any data files then the DBA is given a warning message and the subject is not deleted.

(m) Add Element to File

This function causes the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE ELEMENT.

1. ELEMENT NAME \_\_\_\_\_
2. PARENT FILE NAME \_\_\_\_\_
3. PARENT DATA BASE NAME \_\_\_\_\_

The three input items will all be character strings. The parent file name and the parent data base name are used by the processor to locate the record which describes the parent file. Once this record is located the processor will delete the input element from the list of file elements.

(n) Add Subject to File

This function causes the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE SUBJECT.

1. SUBJECT NAME \_\_\_\_\_
2. CONTAINING FILE NAME \_\_\_\_\_
3. CONTAINING DATA BASE NAME \_\_\_\_\_

The function performs an operation identical to the prior function except that a subject is added to the list of subjects instead of an element to the list of elements.

(o) Delete Element From File

This function causes the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE ELEMENT.

1. ELEMENT NAME \_\_\_\_\_
2. PARENT FILE NAME \_\_\_\_\_
3. PARENT DATA BASE NAME \_\_\_\_\_

The function will locate the record containing the element by using the parent data base name to locate the appropriate File File and then use the parent file name to locate the record within the file. The element is then deleted from the list of file elements.

(p) Delete Subject From File

This function causes the following display.

ENTER THE FOLLOWING INFORMATION ABOUT THE SUBJECT

1. SUBJECT NAME \_\_\_\_\_
2. CONTAINING FILE NAME \_\_\_\_\_
3. CONTAINING DATA BASE NAME \_\_\_\_\_

As in the previous function, the processor will locate the record in the proper File File describing the file which contains the subject. The subject will then be deleted from the file's subject list.

(q) Display Information

This function causes the following selection menu to be displayed.

SELECT TYPE OF DISPLAY:

1. DISPLAY ALL DATA BASES IN NAD
2. DISPLAY DATA BASE DESCRIPTION
3. DISPLAY ALL FILES CONTAINED IN DATA BASE
4. DISPLAY FILE DESCRIPTION
5. DISPLAY ALL ELEMENTS CONTAINED IN A FILE
6. DISPLAY ELEMENT DESCRIPTION
7. DISPLAY ALL SUBJECTS REFERENCED BY A FILE
8. DISPLAY ALL FILES CONTAINING ELEMENT
9. DISPLAY ALL FILES PERTAINING TO SUBJECT
10. DISPLAY ALL USER ELEMENTS
11. DISPLAY USER ELEMENT DESCRIPTION
12. TERMINATE

ENTER YOUR CHOICE: \_\_\_\_\_



Each of these choices allows the DBA to display particular information about the User NAD. Given all of these displays the DBA should be able to determine the current contents of any portion of the User NAD and thus should be able to determine the need for any updates and the validity of any changes made. The function of each of these displays is explained below:

1: Display all Data Bases in NAD

This display lists all Host Data Bases which have been defined in the User NAD.

2: Display Data Base Description

This display requests a data base name and then displays all the information contained in the NAD about that particular data base. The information will include the data base name, the associated data base management system, and a text description of the data base.

3: Display All Files Contained in Data Base

This display requests a data base name from the user. It uses this name to list all files contained in the data base as defined in the User NAD.

4: Display File Description

This display requests a file name and the parent data base name to be used in identifying the file. Once the file is located the file name and the text description of the file will be displayed to the User DBA.

5: Display All Elements Contained in File

This display requests a file name and the parent data base name to be used in identifying the file. Once the file is located a list of all elements contained in the file is displayed to the User DBA.

6: Display Element Description

This display requests an element name. The element name is used to locate the record in the Standard Element file. The information which is then displayed to the DBA is the element name, the text description, the element type and the element unit of measurement.

7: Display All Subjects Referenced by a File

This display requests a file name and the parent data base name. This information is used to locate the file and then the list of all subjects referenced by the file is displayed to the User DBA.

8: Display All Files Containing Element

This display requests an element name. A list is then displayed showing all files which contain the element.

9: Display all Files Pertaining to Subject

This display requests a subject name. It then displays all files which cover the subject.

10: Display All User Elements

This display lists all user elements contained in the User NAD.

11: Display User Element Description

This display requests a user element name. The name is used to locate the appropriate record in the User Element file. The processor then displays the user element name, the standard element name, the value conversion algorithm identifier, the user element type, and the user element unit of measurement.

12: Terminate

The terminate function will return from the display selection menu to the User DBA transaction selection menu.

b. User Data Description Processor Design

The User Data Description Processor consists of a set of modules each responding to a particular User DBA request. The level of nesting for most of the modules is two with the exception being the display modules which are at level three. The two major functions performed by the processor are maintenance of a data base (User NAD) and generation of a set of displays. The display generation and input interface is performed by use of the Terminal Transparent Display Language (TTDL). The data base interface is performed via a set of predefined subroutines which are yet to be written. These routines can be written to interface to any

DBMS which is eventually chosen to meet TIIN requirements. A description of TTDL may be found in the User's Manual for SSB Release III. A description of the DBMS interface subroutine may be found in Appendix E of this report. A thorough description of the functions and operations of each of the User Data Description Processor may be found in the TAP System/Subsystem Specifications or the TAP Program Documentation.

#### 4. HOST DATA DESCRIPTION PROCESSOR

The Host Data Description Processor guides the Host Data Base Administrator through the process of describing that portion of his particular host data base which will be made available to the network. Any data files or data elements not described through this process will not be available to TIIN analysts at any of the user nodes. The data description process is accomplished in an interactive session. The processor prompts the DBA with a selection menu displaying the types of modifications which may be made and informational displays which may be flashed. The DBA selects his choice and the processor then prompts for the appropriate input. Once the change has been accomplished the selection menu is again flashed and the process is repeated. This sequence is continued until the DBA has finished making changes and requests that the session be terminated.

##### a. Host DBA Capabilities

##### (1) General Description

In constructing the Host NAD the DBA describes two distinct characteristics of each host data file added to the network. The first characteristic is the structure of the data file as contained in the host data base. This includes a description of each of the elements contained in the file and the specification of an algorithm for converting the element's standard representation into its equivalent host representation. The second characteristic described by the DBA is the structure of the response file. Each host data file has associated with it a response file structure. The structure is defined in the Host NAD. This definition is used by the Response Normalizer to generate a response file. The normalized response file is then manipulated by TIIN modules in operations such as response file merger and user report generation. For more detailed information concerning the layout of response files or the workings of the Response Normalizer see the TIIN Response Normalization, Final Technical Report.

The description of the data file characteristics is performed before the description of the response file characteristics. The Host DBA need only follow one rule when defining a data file: Define the data file before defining any of its elements. The elements may be defined in any



order. Any elements the DBA chooses not to define will not be available to any TIIN analyst. The same holds true for host data files. Any file the DBA chooses not to define will not be available to any TIIN analyst.

The Response Normal Format must be understood by the Host DBA before he is able to define the response file characteristics. A detailed discussion of the Response Normal Format is found in the TIIN Response Normalization, Final Technical Report. The basic concept, however, is that a file consists of a collection of records. Each record consists of a set of hierarchically related segments (i.e., a n-ary tree with each segment represented by a node and each segment's parent segment represented by the node's immediate ancestor in the tree). Each segment consists of a list of directly related elements. The Host DBA must determine the hierarchical relationship between groups of elements (segments) in the file. Each segment is then defined and the parent segment as determined by the hierarchical relationship is specified. The member elements of each segment are then specified. When this process is completed the response file format has been completely defined. In summation, the rules which must be adhered to when defining a response file are as follows.

- o Define the structure of the host data file before defining the structure of the response file
- o Define the response file before defining any of its segments
- o Define the segment before defining any of its elements
- o Define an element as a host data file element before defining it as a response file element.

## (2) Specific Capabilities

When the DBA activates the User Data Description Processor he will be flashed the following selection menu.

SELECT TRANSACTION TYPE:

- |                                   |                                  |
|-----------------------------------|----------------------------------|
| 1. DEFINE NAD                     | 2. DEFINE FILE                   |
| 3. DEFINE RESPONSE FILE SEGMENT   | 4. DEFINE ELEMENT                |
| 5. DEFINE RESPONSE FILE ELEMENT   | 6. DELETE NAD                    |
| 7. DELETE FILE                    | 8. DELETE RESPONSE FILE SEGMENT  |
| 9. DELETE ELEMENT                 | 10. DELETE RESPONSE FILE ELEMENT |
| 11. MODIFY ELEMENT                | 12. DISPLAY FILE                 |
| 13. DISPLAY RESPONSE FILE         | 14. DISPLAY ELEMENT              |
| 15. DISPLAY RESPONSE FILE SEGMENT | 16. TERMINATE SESSION            |

ENTER YOUR CHOICE \_\_\_\_\_.

Each of these functions will either allow the DBA to modify the NAD or else display its current contents. When finished with his changes the DBA selects the "terminate session" command to terminate the Host Data Description Processor. A description of each of the available functions follows.

(a) Define NAD

This function creates a Host NAD and thus is normally entered only once per Host Data Base. It creates the Element File and the RNF Descriptor File which comprise a Host NAD. The function requires no DBA input because no data is entered into either file.

(b) Define File

This function adds a new host file to the NAD and thus makes it part of the scope of information of TIIN. The following prompt is flashed to the DBA.

ENTER NAME OF FILE: \_\_\_\_\_

The file name is used to add a record to the Element File. The file is now defined as a host data file and as a response file. Still to be defined are the data file elements and the response file structure.

(c) Define Response File Segment

This is the function used to define the hierarchical relationship between segments in the response file. The following display is flashed to the DBA.

ENTER THE FOLLOWING INFORMATION ABOUT THE SEGMENT.

1. NAME OF FILE CONTAINING SEGMENT \_\_\_\_\_
2. SEGMENT ID \_\_\_\_\_
3. PARENT SEGMENT ID \_\_\_\_\_

Note that the parent segment identifier is used to determine the structure of the hierarchical tree representing the response file. It is also important to note that segment identifiers only uniquely determine the segment type within a single host file. No correlation is implied between segments with identical identifiers in different host files.

(d) Define Element

This function adds an element definition to the definition of a host data file. The following display requesting input is flashed to the DBA.

ENTER THE FOLLOWING INFORMATION ABOUT THE ELEMENT.

1. NAME OF FILE CONTAINING ELEMENT \_\_\_\_\_
2. TIIN ELEMENT NAME \_\_\_\_\_
3. HOST ELEMENT NAME \_\_\_\_\_
4. HOST ELEMENT TYPE \_\_\_\_\_
5. HOST ELEMENT UNIT OF MEASUREMENT \_\_\_\_\_
6. SPECIAL CHARACTERISTIC WORD \_\_\_\_\_
7. TIIN TO HOST VALUE CONVERSION ALGORITHM ID \_\_\_\_\_

The TIIN element name is the standard element name or the name which is standard throughout the network. Its scope is global throughout all host data bases available through TIIN. All host attributes (i.e., element name, element type, element unit of measurement) are assumed to be local to the host file containing the elements. It is the responsibility of the Host DBA to map his local elements to their standard equivalents. The



special characteristic word structure varies from one data base to another. It defines those characteristics of an element needed by TIIN to generate a query against it (e.g., in DIAOLS the fact that an element is a relational periodic element would be maintained since these require special query verbs). The TIIN-to-Host value conversion identifier specifies the value translation mechanism to be used in converting the standard element representation into the host element representation.

(e) Define Response File Element

This function is used after the response file segments have been defined to add the proper elements to each segment. Each time an element is added to a segment, certain characteristics about the element are defined. The display below is the prompt to the DBA to enter these characteristics.

ENTER THE FOLLOWING INFORMATION ABOUT THE ELEMENT.

1. NAME OF FILE CONTAINING ELEMENT \_\_\_\_\_
2. TIIN ELEMENT NAME \_\_\_\_\_
3. ID OF SEGMENT CONTAINING ELEMENT \_\_\_\_\_
4. TIIN ELEMENT TYPE \_\_\_\_\_
5. TIIN ELEMENT UNIT OF MEASUREMENT \_\_\_\_\_
6. HOST TO TIIN VALUE CONVERSION ALGORITHM ID \_\_\_\_\_

The TIIN element name, type, and unit of measurement are used because this is a standardized response format rather than a response format associated with a particular host data base. The segment identifier is local to the host file. When merging two or more response files the merger algorithm must use standard element names, rather than segment identifiers, to compare records since only they are global. The Host-to-TIIN value translation mechanism used in converting host values into equivalent standard values.

(f) Delete NAD

This function is used to delete an entire Host NAD and thus an entire Host Data Base from the realm of consideration of TIIN. Once the Host NAD has been deleted the Host Data Base is unavailable to any TIIN analyst.

(g) Delete File

This function causes a host file to be deleted from the Host NAD and thus from the realm of knowledge of TIIN. The function requests only the file name from the DBA.

(h) Delete Response File Segment

This function deletes a single segment descriptor from the response file descriptor. Before this operation may be performed all response file elements must be deleted from the segment.

(i) Delete Element

This function deletes a single element from the host data file description. The element may no longer be queried from this host file.

(j) Delete Response File Element

This function is used to delete an element description from the response file description. Once this is performed the element will no longer be returned from this host file.

(k) Modify Element

This function allows the DBA to modify the description of a Host element without first deleting it from the NAD and then redefining it.

(l) Display File

This function is used to display the list of element descriptions contained in the description of a host data file. The DBA is asked to input the file name of interest.

(m) Display Resonse File

This function displays the set of segment descriptors which are currently defined for a particular response file descriptor. The analyst may use this display to determine the current hierarchical organization of the response file. The only input required by the function is the response file name.

(n) Display Element

This function requires that the DBA specify the host element name and the parent file name. The processor then displays all information contained in the NAD about the element.

(o) Display Response File Segment

The function displays all elements defined as part of a segment. The DBA must input the file name and the segment identifier. This function when combined with the "Display Response File" function presents a complete picture of the structure of the response file to the DBA.

(p) Terminate Session

The function is used to terminate the interactive session with the Host Data Description Processor.

b. Host Data Description Processor Design

The Host Data Description Processor consists of a set of modules, each performing a particular function as defined in the DBA function selection menu. The processor performs the Host NAD update functions by means of a set of predefined subroutines which can be written to interface with any DBMS chosen for use by TIIN. A description of these routines may be found in Appendix E of this report. The processor performs its display function by means of TTDL calls. These are described in the User's Manual for SSB Release III. A detailed description of the processors modules may be found in both the System/Subsystem Specifications for TAP and the Program Documentation for TAP.

5. FUTURE DEVELOPMENT

a. Interactive Capability

The Analyst Aids Processor, the User Data Description Processor and the Host Data Description Processor are all interactive processors driven primarily by user menu selection. This works well for an inexperienced user who needs a list of choices to perform his functions but as his experience level increases the continual redisplay of a menu becomes tedious. These interactive processors should adapt to this situation and only display lists of choices when requested. They should also provide to the experienced user the capability to enter a whole string of parameters at once instead of requiring him to wait for the prompt for each. The philosophy is to only provide the level of support



required by the analyst. This allows the experienced analyst to proceed through the process almost as quickly as he can enter data while providing the inexperienced user with as much support as required.

b. Automatic Verification of User DBA Updates

Each time the User DBA defines a new host data base, data file, or data element as being available at his node he is assuming it is already defined within the appropriate Host NAD. The User Data Description Processor should verify the validity of that assumption before allowing the change in the User NAD to be completed. This would require the processor to have the capability of communicating with all nodes containing a Host NAD.

c. Automatic Notification of Host NAD Updates

A mechanism for the timely dissemination of Host NAD update information to each affected User NAD should be developed. Several known alternatives presently exist for implementing this improvement. One alternative is for each host to periodically broadcast to all User NADs standard descriptions of any updates which have occurred since the last broadcast. Those User NADs which are affected by the change may then be updated. Another alternative is that instead of broadcasting the changes, only the fact that a change has occurred is made known. All User NADs which may be affected by the change may then request all information about the update. A third alternative is that all changes to Host NADs be logged and periodically the User NADs request copies of the logs for the Host NADs which affect them.

## SECTION IV

### QUERY LANGUAGE PROCESSOR

#### 1. INTRODUCTION

The Query Language Processor (QLP) is the TIIN/TAP module responsible for translating the Transparency Examples Language (TEL) into the Data Request Intermediate Format (DRIF). TEL is the TIIN user query language. DRIF is the internal query format before host file selection or user element name translation have taken place. This section describes both the TEL and the DRIF query formats before describing the QLP.

#### 2. TRANSPARENCY EXAMPLES LANGUAGE

The Transparency Examples Language (TEL) is an example of a simple user query language. TEL is used by the intelligence analyst to submit queries to TIIN. It has been designed to provide access to the currently designed features of TIIN as well as to provide a base that may be readily added to. As features such as multifile queries and user specification of partial query merging operations are developed the TEL language can be expanded to include them. It is currently aimed at querying the DIAOLS system. It should be noted by the reader that TEL is a sample query language only. It has been designed to demonstrate the feasibility of a common query language. It has been used in TAP to verify the logic of the Query Language Processor. In a real-time TIIN system any other suitable query language or combination of languages could be used.

##### a. Language Syntax

The syntax description of TEL appears in Figure IV-1. Each TEL query consists of up to four different selection phrases. The host selection phrase and the file selection phrase are optional to any query and both may be omitted. These two phrases select the host computer and the specific file to which the query is directed. If either phrase has not been included within the query the TIIN system will select the appropriate host or file at which to direct this query. The record selection phrase is required and allows the user to specify the record selection criteria to be used in selecting qualifying target records. The output selection phrase is also required. It specifies which fields within each record are to be displayed as output.

##### (1) File Selection Phrase

A file selection phrase is necessary if a host file is not uniquely determined by the element names used in the analyst query. A file selection phrase consists of the word

<query>	::=	[HOST <host-name>;] [FILES <file-list>;] RETRIEVE <select-expr>; SHOW <element-list>; [RESPONSES <max-response-size>;]
<host-name>	::=	string
<file-list>	::=	<file name> [, <file-list>]
<file-name>	::=	string
<element-list>	::=	<element-name> [, <element-list>]
<element-name>	::=	<std-el-id>   <host-el-id>   <user-el-id>
<std-el-id>	::=	string
<host-el-id>	::=	/string/
<user-el-id>	::=	@string@
<max-response-size>	::=	number
<select-expr>	::=	<and-expr> [OR <select-expr>]
<and-expr>	::=	<with-expr> [AND <and-expr>]
<with-expr>	::=	<not-expr> [WITH <with-expr>]
<not-expr>	::=	[NOT] <paren-expr>
<paren-expr>	::=	(<select-expr>)   <criteria>
<criteria>	::=	<expr> <relational-op> <expr> [<location>] <geographic-op> <geographic>
<expr>	::=	[<quantifier>] <element-name> [<element-value>]
<element-value>	::=	'string'   number
<quantifier>	::=	EVERY   SOME   NO
<relational-op>	::=	LT   LE   EQ   GE   GT   NE   CONTAINS
<geographic-op>	::=	INSIDE   OUTSIDE   ALONG
<location>	::=	<latitude-el-id> <longitude-el-id>
<latitude-el-id>	::=	<element-name>
<longitude-el-id>	::=	<element-name>
<geographic>	::=	<circle>   <route>   <polygon>
<circle>	::=	CIRCLE (radius, <coordinates>)
<route>	::=	ROUTE (count, radius, <coordinate-list>)
<polygon>	::=	POLYGON (count, <coordinate-list>)
<coordinate-list>	::=	<coordinates> [, <coordinate-list>]
<coordinates>	::=	latitude longitude

Figure IV-1. Syntax Description of TAP Common Query Language



"FILES" followed by one or more file names separated by commas and terminated by a semicolon. For example:

FILES BIO3, TACK78;

The file selection phrase affects all queries entered by the analyst until the next file selection phrase is entered.

To de-select the files the analyst may enter the word "ALL" in place of a list of file names in a file selection phrase, as follows:

FILES ALL;

File de-selection has the effect of allowing file selection to be implicitly determined by the TIIN system based on the element names which appear in queries. This effect applies to all queries entered subsequent to the file de-selection phrase until the next file selection phrase is entered.

## (2) Host Selection Phrase

For the prototype TIIN the analyst is restricted to one host DBMS. The analyst may explicitly select which host is to be queried by entering the host selection phrase consisting of the word "HOST" followed by the host name and terminated by a semicolon. For example:

HOST DIAOLS;

The host selection phrase affects all queries entered by the analyst until the next host selection phrase is entered. The host selection phrase is necessary if the analyst directly uses host element names in the query instead of TIIN network standard element names. Also, the host selection phrase is necessary if the host is not uniquely determined by the element names used in the analyst query.

To de-select the host the analyst may enter the word "ALL" in place of a host name in a host selection phrase, as follows:

HOST ALL;

Host de-selection has the effect of allowing host selection to be implicitly determined by the TIIN system based on the element names which appear in queries. This effect

applies to all queries entered subsequent to the host de-selection phrase until the next host selection phrase is entered.

### (3) Record Selection Phrase

The record selection phrase begins with the keyword RETRIEVE and is followed by a complex boolean combination of element criteria. The element criteria may be combined using parentheses, the boolean operators NOT, AND, OR, and the special operator WITH which is a special version of the AND operator including implicit relationships due to the hierarchical dependencies between host data elements. In a boolean expression the precedence of the operators from highest to lowest is NOT, WITH, AND, OR.

Element criteria may be an arithmetic comparison, a text scan, or a geographic inclusion test. A basic arithmetic comparison consists of an element name, an arithmetic comparison (EQ, NE, LT, GE, GT) and a constant. A basic text scan consists of an element name, or substring operator (CONTAINS), and a constant. A geographic inclusion test consists of an element pair specifying a latitude element and a longitude element, a geographic operator (INSIDE, OUTSIDE, ALONG), and a geographic region specifier (CIRCLE, ROUTE, POLYGON) with its associated parameters.

#### (a) Element Names

The analyst has the option of using element names from the user frame of reference, the network frame of reference, or the host frame of reference. The syntax for network standard element names is as follows:

- o First character must be alphabetic (A through Z)
- o Subsequent characters must be alphabetic (A through Z) numeric (0 through 9), or dash (-)
- o The element name must not coincide with any reserved word in the TIIN query language
- o The maximum number of characters is 255.

EXAMPLES: MISSION-DATE, A0001

The syntax for host element names is identical to that required by the specific host, with only one additional rule. Specifically, a slash (/) must appear immediately before and after the character string which denotes the host name. The slashes serve only to delimit the host element name and are not part of the element name when the query is presented to the host DBMS. A double-slash may be used to represent a single slash embedded in a host element name.

EXAMPLES: /MISSION.DATE/, /0001/, /S,/360/

The syntax for user element names is unrestricted, with an at-sign (@) appearing immediately before and after the character string which denotes the user element name. The at-signs serve only as delimiters and are not part of the user element name itself. A double at-sign may be used to represent a single at-sign embedded in a user element name.

These naming conventions apply primarily in the context of a query entered via the TAP common query language.

#### (b) Element Values

The types of constants, numeric and character strings, may appear in queries entered via the TAP common query language. The syntax for a character string is as follows.

- o The maximum number of character is 255.
- o The string must conform to the restrictions which apply at the host
- o A quote mark (') must appear immediately before and after the character string. The quote marks serve only to delimit the string and are replaced at the host by the appropriate punctuation. Double quote marks may also be used to represent a single quote mark embedded in the character string.

EXAMPLES: 'US'  
'12/17/79'  
'DIDN"T'



The syntax for a numeric constant is as follows:

- o The maximum number of characters is 255.
- o The initial character must be numeric (0 through 9)
- o The numeric constant is delimited by a blank or other special character
- o The string must conform to the restrictions for character strings or numeric constants which apply at the host.

EXAMPLES: 500.3  
12/17/79  
18:01

#### (4) Output Selection Phrase

The output selection phrase consists of the keyword SHOW followed by a list of element names to be displayed. Each element name selects one field from each of the output records to be displayed. The names may be either TIIN standard names, user names or host names as specified in the element name section above. Only one output selection phrase per query is allowed. If SHOW ALL; is specified, all fields will be displayed.

#### (5) Response Size Phrase

This optional phrase consists of the keyword RESPONSES followed by a number. The phrase specifies the maximum number of query responses which the user wants returned to his site.

#### b. Sample Queries

Suppose a user wishes to query the "COUNTRIES" file for the capitols and populations of all countries having an Armed Forces strength of greater than 50,000 men and located within a circle of 2000 miles from Saigon. The query would look like the following:

FILES COUNTRIES;

RETRIEVE MILITARY-STRENGTH GT 50000  
AND COUNTRY-LOCATION INSIDE CIRCLE (2000, 1000N  
1070000E);

SHOW CAPITOL, POPULATION;

Another possibility is that we wish to interrogate host "A1" to determine which records in the "CITIES" file are not within 3000 miles of Moscow nor within 3000 miles of Washington. We wish to list all fields of each record meeting these qualifications.

HOST A1;

FILE CITIES;

RETRIEVE NOT (CITY-LOCATION INSIDE CIRCLE (3000, 560000N 380000E) OR CITY-LOCATION INSIDE CIRCLE (3000, 390000N 770000W));

SHOW ALL;

In this example the record selection phrase could be rewritten as:

RETRIEVE CITY-LOCATION OUTSIDE CIRCLE (3000, 560000N 380000E)  
AND CITY-LOCATION OUTSIDE CIRCLE (3000, 390000N 770000W);

### 3. DATA REQUEST INTERMEDIATE FORMAT

#### a. Functional Design

The DRIF (Data Request Intermediate Format) is the data structure used for passing queries from the QLP (Query Language Processor) to the User QIP (Query Intermediate Processor). A single DRIF is produced by the QLP for each analyst query.

The DRIF is designed to be general and capable of handling a superset of all features appearing in typical intelligence query languages. To allow this, the DRIF design is open-ended and can easily assimilate additions. The verbs and operators in DRIF are system operators, allowing the addition of new operators without any changes to the DRIF structure.

#### (1) DRIF Structure

The DRIF consists of two kinds of information. General information about the query, such as user identifier, query identifier, and other fixed information are stored in the fixed portion of DRIF. The query itself is stored in reverse Polish notation in the conditional table in the variable length portion of DRIF.

## (2) Data Sources

The information contained in the DRIF is derived from two sources: the user/analyst and the TIIN system. System provided information includes the user node name, the originating terminal identifier, and the time the query originated. Analyst-provided information includes data selection criteria, a list of data names for which data values are wanted, and the maximum number of responses wanted.

In order to account for all queries, the TIIN system will assign a unique identification to each query. The identification will consist of the user node at which the query originated, the user ID, and the time the query was originated.

The analyst may also provide information to locate the data. The analyst is permitted to specify a host data base and, optionally, the name of a file located at the data base. When the name of the data base is not specified the TIIN system will try to satisfy the query from all the data bases on the network. When the data base name is specified and the file name is not, the TIIN system will try to satisfy the query from the data located at the specified data base.

## (3) Language Independence

The structure of the DRIF is independent of the language in which the query was originally stated. The syntax of the original language is translated into the DRIF syntax, and all functional names are translated into standard TIIN function codes at the time the DRIF is generated. Consequently, in order to interpret a DRIF, TIIN does not need to know anything about the source language of the query.

## (4) Data Base Dependence

The only parts of the DRIF not in the standard TIIN system frame of reference are the element names and element values. The module generating the DRIF is familiar with the query language it is translating, but is not familiar with the elements, since data elements are not part of the language. Different users in the system may use the same query language but different sets of elements and different names for the same elements. The QNF Generator, the module that transforms the DRIF into the QNF, has access to element name and value translation tables in the NAD (Network Access Directory) and therefore is the logical module to translate names and values from the user to the TIIN standard frame of reference.



The element names appearing in the DRIF can be divided into three categories: user names, TIIN standard names, and host names. User names are names that are known at the local node and that can be translated into TIIN standard names by using translation tables in the NAD. Host names are names local to the host DBMS and therefore not subject to the translation process. When an analyst specifies any host names in his query, he is required to also specify the host data base.

b. DRIF Layout

The DRIF can be divided into two sections: the header and the conditional table. The header contains information which TIIN uses to process the user's data request. The conditional table contains the TIIN translation of the user's data request.

(1) Header

The structure of the DRIF and QNF is shown in Figure IV-2. It should be noted that some of the items in the header are only determined once the DRIF has been translated into QNF by the User QIP. The following is a description of the header items. The only item in this list which is generated directly by the QLP is the maximum number of responses.

- Word 0      USER NODE ID is the network identification of the node from which the query originated.
- Word 1      ORIGINATING USER IDENTIFIER is the TIIN identification of the user/analyst who initiated the data request. This ID is unique within a node.
- Words 2-3      TIME QUERY ORIGINATED is the date and time that the DRIF for the data request was constructed. Words 0-3 together constitute a unique identifier for the user request (DRIF).
- Word 4      QNF IDENTIFIER is used to differentiate between the different QNF's originating from the same DRIF.
- Words 5-6      FLAG WORDS are status words used to indicate the operational status of the message headed by this block.
- Word 7      ORIGINATING TERMINAL IDENTIFIER is the TIIN identification of the terminal which initiated construction of the data request.

WORD	
0	USER NODE IDENTIFIER
1	ORIGINATING USER IDENTIFIER
2	TIME QUERY ORIGINATED
3	
4	QNF IDENTIFIER
5	FLAG WORDS
6	
7	ORIGINATING TERMINAL IDENTIFIER
8	SECURITY CLASSIFICATION
9	MAX NUMBER OF RESPONSES
10	HOST DATA BASE IDENTIFIER
11	FORMAT CODE
12	COUNT OF RETURNEES (=R)
13	RETURNEES (1 WORD EACH) ⋮
13+R	CONDITIONAL TABLE LENGTH (BYTE COUNT)
14+R	CONDITIONAL TABLE ENTRY 1 ⋮ ⋮ CONDITIONAL TABLE ENTRY M
	CONDITIONAL TABLE

Figure IV-2. Layout of DRIF and QNF

- Word 8 SECURITY CLASSIFICATION is a one-word code indicating the classification of the user/analyst originating the data request.
- Word 9 MAX NUMBER OF RESPONSES is the maximum number of records the user/analyst wants to retrieve.
- Word 10 HOST DATA BASE IDENTIFIER is the network identification of the host DBMS which will process the user/analyst query. A blank value in this field in the DRIF received from the user language processor (TAP) will be replaced with a valid host identification by the host selector module.
- Word 11 FORMAT CODE is the TIIN identification for the user specified response format.
- Word 12 COUNT OF RETURNEES is the number of users to receive responses.
- WORDS RETURNEES is a list of TIIN one word user identifiers of users, other than the original user, who are to receive responses.

## (2) Conditional Table

The entries in the conditional table are variable length and have been entered in the order of Reverse Polish Notation by the ALP. Each entry includes a byte count, a type code and either a string of bytes or a keyword identifier depending upon the entry (See Figure IV-3).

### (a) Entry Byte Count

This is a one byte count of the number of bytes comprising this entry. The count includes this byte. The minimum entry byte count is three.

### (b) Entry Type

This is a one byte code which specifies the type of the entry. The current list of type codes is:

- 1 = Verb
- 2 = Operator
- 3 = User Element Name
- 4 = Standard Element Name
- 5 = Host Element Name
- 6 = User Element Value
- 7 = Standard Element Value



<RPN-expression>	::=	<multi-db-query> complete-QNF-identifier multi-db-query
<multi-db-query>	::=	<single-db-query> <single-db-query> <multi-db-query> <response-operator>
<single-db-query>	::=	<phrase-list> <phrase-list> <file-list>
<file-list>	::=	file-specification file-specification <file-list>
<response-operator>	::=	MERGE   JOIN   DIFFERENCE
<phrase-list>	::=	<verb-phrase> <phrase-list> <verb-phrase>
<verb-phrase>	::=	<select-phrase> <print-phrase>
<select-phrase>	::=	<select-expr> SELECT
<select-expr>	::=	<criteria> <select-expr> <unary-logical> <select-expr> <select-expr> <binary-logical>
<criteria>	::=	<geographic-expr> <relational-expr>
<geographic-expr>	::=	<expression> <geographic> <geographic-op>
<geographic>	::=	<circle> <route> <polygon>
<geographic-op>	::=	INSIDE   OUTSIDE
<circle>	::=	radius <coord> CIRCLE
<route>	::=	coord-count half-width <node-list> ROUTE
<polygon>	::=	coord-count <node-list> POLYGON
<coord-list>	::=	<coord> <coord> <coord-list>
<coord>	::=	latitude longitude
<relational-expr>	::=	<expression> <expression> <relational-op>
<expression>	::=	element-name element-value element-name <quantifier>
<relational-op>	::=	GE   GT   LE   LT   EQ   NE   HAS   HASNOT
<quantifier>	::=	ALL   NO
<binary-logical>	::=	AND   OR   WITH
<unary-logical>	::=	NOT
<print-phrase>	::=	<element-list> PRINT
<element-list>	::=	element-name element-name <element-list>

Figure IV-3. Syntax Description of the Reverse Polish Notation (RPN) Used in the Conditional Table of the QNF.

- 7 = Standard Element Value
- 8 = Host Element Value
- 9 = File Name
- 10 = Response Operator
- 11 = Complete QNF Identifier
- 12 = Host Name
- 13 = Maximum Number of Target Records to be Returned

(d) String

This field normally contains a variable length character string representing a name or value. The only two exceptions to this rule are when the entry type equals one or zero. If the entry type equals zero the entry is a verb and the string field contains a single byte integer verb code (0 = SELECT, 1 = PRINT). If the entry type is one the entry is an operator and the integer code representing it is found in Figure IV-4.

1 Verbs

In the TIIN prototype the verb code differentiates between the select statement and the print statement with only one select and one print code per query. In later stages of development the verb codes may include data manipulation verbs, different kinds of retrieval and output verbs, as well as mixed sequences of verb phrases.

a SELECT (Verb Code: 1)

SELECT is a verb associated with the reverse Polish expression which qualifies the records to be extracted from the data base. The operators associated with the SELECT verb are described below.

b PRINT (Verb Code: 2)

The expression associated with PRINT specifies the elements to be extracted from the data base as a result of this query. In the TIIN prototype, the expression associated with PRINT is a list of element names, and cannot contain any operators.

Operator/Functions	TIIN Standard Code	Number of Operands
<u>Logical</u>		
AND	1	2
OR	2	2
NOT	3	1
WITH	4	2
<u>Relational</u>		
GT (greater then)	10	2
GE (greater than or equal)	11	2
LT (less than)	12	2
LE (less than or equal)	13	2
EQ (equal)	14	2
HAS	15	2
HASNOT	16	2
INSIDE	17	2
OUTSIDE	18	2
	19	2
<u>Geographic</u>		
CIRCLE	20	3
ROUTE	21	variable (min. 6)
POLYGON	22	variable (min. 7)
<u>Quantifiers</u>		
ALL	30	1
NO	31	1
<u>Response</u>		
MERGE	40	2
JOIN	41	2
DIFFERENCE	42	2

Figure IV-4. Standard Operator Codes in DRIF

## 2 Operators Used with the SELECT Verb

The operators defined for the TIIN prototype, and their associated TIIN standard operator codes are shown in Figure IV-4. These codes are used in the DRIF.

### a Logical Operators

The logical operators, AND, OR and WITH must each have two operands. The logical operator NOT must have one operand. All operands associated with logical operators must generate boolean values when evaluated. An operand of a logical operator must therefore be an expression formed from either another logical operator, a relational operator, a geographic operator, or any other operator resulting in a boolean value.

The AND and WITH operators have the same meaning unless a hierarchy of record segments is involved. In a hierarchy where several subordinate segments have the same parent segment, the AND operator indicates the criteria may be satisfied in different subordinate segments under the common parent, whereas the WITH operator indicates the criteria must be satisfied in the same subordinate segment.

### b Relational Operators

Seven relational operators (GT, GE, LT, LE, EQ, NE, HAS, HASNOT) can be specified in the QNF. Each relational operator must have two operands. If one operand is an element name and the other one is a value or an expression, the element name must be specified as the first operand (i.e., 5 GT A would be converted to A LT 5). The first six operands specify the standard six relationships. The operator HAS is used to select element values which include the specified partial value. HASNOT selects those values which do not include the specified partial value.

### c CIRCLE

Opernads: Radius, latitude, longitude



The CIRCLE operator tests whether the unit described in the current record in the data base is inside or outside of a specified circle, and returns a boolean result. The operator must always have three operands. The first operand specifies the radius of the circle in nautical miles. The second and third arguments specify the latitude and longitude of the center of the circle. The latitude and longitude are specified in the following format: DDDMMSSg, where DDD are degrees, MM are minutes, SS are seconds, and g specifies E, W, N or S. All units inside the circle will be retained, unless the function is qualified by a NOT operator in order to retain all units outside the circle.

d     ROUTE

Operands: Number of nodes, half-width, latitude longitude, . . . .

The ROUTE operator tests whether a given location is within a specified route. A route is defined by specifying the half-width distance and the coordinates of each node defining the route. The ROUTE operator contains six operands for the simplest route consists of two nodes, and another two operands for each additional node within the route. The first operand indicates the number of points in the specified route. This number must be greater or equal to two. The second operand contains a number of equal to the half-width of the route, in nautical miles. The remaining operands are pairs of numbers, the first number of each pair specifying the latitude, and the second number specifying the longitude of a node. The number of such pairs must be equal to the number of nodes specified in the first operand. All units inside the route will be retained, unless the function is qualified by a NOT operator in order to retain all units outside the route.

e     POLYGON

Operands: Number of nodes, latitude, longitude . . .

The POLYGON operator tests whether a given set of coordinates specifies a point inside or outside a polygon. The polygon is defined by providing a list of at least three coordinate points. The points must be listed consecutively clockwise or counterclockwise. The Polygon operator requires at least seven operands. The first operand specifies the total number of nodes comprising the polygon, at least three. The remaining operands are pairs of latitudes and longitudes specifying the nodes. The number of latitude and longitude pairs must equal the number of nodes specified in the first operand. All units inside the polygon will be retained, unless the function is qualified by a NOT operator in order to retain all units outside the polygon.

#### f Quantifiers

Quantifiers are a general class of operators applied to repeating elements. When referring to a repeating element name in a query, an analyst actually refers to all the values for that element in one record. Quantifiers allow a user to refer to specific value(s) within a record, such as FIRST, LAST, NO, ALL. The initial set of TIIN standard operators allows only ALL and NO.

#### g Response Operators

Response operators are binary functions specifying the rules for combining multiple responses to a query.

Currently, three such operators are proposed, MERG, JOIN, and DIFFERENCE, corresponding roughly to the OR, AND, and AND NOT logical operators, respectively.

The MERGE operator operates on identical parallel queries and merges identically formatted files into a single file, deleting duplicate entries.

The JOIN operator matches records from the two input files on a one-to-one basis, forming composite records. The records are matched by comparing the values of a specified

set of elements appearing in both files. Records lacking a match in the other file are discarded.

The DIFFERENCE operator matches records from two files in the same manner as the JOIN operator. Only records existing in the first file, but having no corresponding matches in the second file, are kept.

#### 4. QLP CAPABILITIES

The QLP performs two basic functions. Its primary duty is to translate a TEL query into DRIF. This operation basically involves the mapping of TEL operators and TEL verbs into their DRIF equivalents, labeling operand type as either User, TIIN Standard or Host, and rearranging the order of the query so as to achieve reverse Polish ordering. The second function performed is that of aiding the analyst in entering the query. This capability is currently limited in its sophistication but may be enhanced in the future. These two basic functions have divided the QLP into two functional modules. That which interfaces with the analyst and aids him in query submission is called the Interface Module. That which translates the query from TEL into DRIF is called the Translator Module.

##### a. Interface Module

Initially the Interface Module will be limited in its capabilities. These functions will be immediately implemented.

- o Display the basic query submission format to the analyst including the usage of each of the basic phrases. This will not be a display of the entire syntax of TEL.
- o Accept a TEL query from the user and activate the translator module to perform TEL to DRIF Translation.
- o Terminate an analyst's query session at his command.

The first two functions may be repeated as many times as the analyst requests while the third will terminate a session. The improvements to the Interface Module which are perceived at the present include:

- o Query Storage Capability - Allows the analyst to save his query in a file so that he may later retrieve it.

- o Query Retrieval Capability - This allows the analyst to retrieve any query saved as a result of a save operation and submit it to the system. Note that the query will be translated each time it is submitted because the source is saved, not the DRIF representation.
- o Query Deletion Capability - This allows an analyst to delete any query from the save file.
- o Edit Capability - This allows the analyst to modify existing queries instead of having to reenter the query every time a change is needed.
- o Syntax Description of Query Language - This feature would be an expansion of the immediately implemented display function. It would describe in more detail the query language syntax and, in particular, provide a detailed description of each of the retrieval phrase operators.

Additional analyst aid functions may be added as the need for them arises. The Interface module is structured so as to easily accommodate the addition of these functions.

#### b. Translator Module

The Translator Module is responsible for receiving a query from the Interface Module, translating it into its DRIF representation, flagging any errors it finds and, if no fatal errors are found, passing the DRIF onto the Host QIP. As was the case with the Interface Module, the Translator Module has been designed with flexibility in mind, especially in the area of future additions and modification. Some of the key points in the design include:

- o Table Driven Algorithms - This allows for the simple addition or deletion of keywords of the language which will be classified as to type (i.e., verb, operator, quantifier, geographic, etc.) and added to the keyword table. This keyword table is the driving force behind the translator.



- o Top-Down Modularized Design - This will allow for operators to easily be implemented. Once they are added to the **keyword** table, a module will be written to correctly parse that particular operator. The interface to call the new module will already be in existence. This idea applies in reverse for deletion of an operator. In this case the module to parse that particular operator will be deleted along with the entry in the keyword table.
- o Operator Precedence Grammar Within Retrieval Phrase - The set of TEL operators have been assigned precedence values which are specified in the keyword table entry for each operator. This allows the use of simple existing expression parsing techniques. As an operator is added to the language it will be assigned an appropriate precedence value and entered into the keyword table. The module for handling the precedence relationships will already be in existence.
- o Diagnostics - The translator will provide diagnostics to the user which will aid him in correcting his query. Initially these diagnostics will be limited since the major objective of this task is a functioning translator. The interface is there, however, for providing a much more complete set of diagnostics in the future.

In summary, the primary design goal has been a functioning translator for the TIIN system. This goal has been met. Also the hooks for easy expansion should be present. It is felt that as time passes the TEL language will be modified. For this reason the translator has been designed to be easily modified.

## 5. QLP DESIGN

### a. Interface Module

The Interface Module is very simple in design. It will use TTDL for the terminal interface. As an analysts activates the system the User Interface Module will call TTDL to display a select menu. This menu will allow the analyst to choose from the following sets of commands.

- o DISPLAY QUERY FORMAT
- o ENTER QUERY
- o TERMINATE SESSION

Once the user has made a selection, TTDL will pass the selection number to the User Interface Module. This number will be used to select the appropriate submodule which will handle the request. New functions may be easily added by adding an entry to the selection menu and adding the submodule to perform the function.

b. Translator Module

The proper functioning of the Translator Module is necessary to get the prototype TIIN system up and running. The module will primarily be driven by the keyword table. Each dedicated symbol and keyword is contained in this table so that the authenticity of any given symbol as a keyword may be quickly checked. A table entry contains the DRIF representation of the symbol making translation a matter of table look up. A type code identifying the class of symbol is also contained in each entry. This makes it simple to select the proper submodule to handle each type of keyword. Also, since the operators have been grouped according to type, fewer modules are needed to handle them. One module will handle multiple operators of any given type. When new operators are added to the language, they will often be of an existing type. In this case all that is necessary for their inclusion in the language is an entry to the keyword table. If the operator is of a new type in addition to its inclusion in the keyword table, a new module will have to be added to handle this new type.

The remaining three data structures used to control translation are the operator stack, the operand stack and the legal next symbol word. The operator stack is used in conjunction with the operator precedence value in the keyword table to determine the proper location in the DRIF polish for each operator. As each new operator is scanned from the TEL query, its precedence is compared with the precedence of the operator at the top of the stack. If the precedence of the newly scanned operator is less than or equal to that of the stack operator the stack operator is popped, checked for correctness of operands and added to the DRIF. The routine to check for correctness is determined by the class code obtained from the keyword table. The new operator will be repeated until the new operator's precedence is greater than that of the top of the stack. At this time it would be pushed onto the stack and processing will continue with another scan of the query text.

When the operator was popped from the stack and before it was added to the DRIF, a check was made to see if it was operating on legal operands. This check is made by popping the appropriate number of operands from the operand stack and checking their type. If this is a relational operator the type of value is marked by noting the type of the associated name. Finally a resultant boolean operand is pushed onto the operand stack and operator processing is continued.

The final data structure used by the Translator is the legal next symbol word. This is a single 16-bit word with each bit representing a class of symbols. Upon entry each symbol handling module will check this word to see if it has the appropriate class bit set. If it does, processing continues. If it doesn't, this is an illegal class of symbol for this particular location in the query and an error is flagged to the analyst. Just before the symbol handling routine returns to the main line parser it will set the appropriate bits in the legal next symbol word. These will be used by the next symbol handling routine to determine whether its set of symbols is legal.

#### (1) Keyword Table Data Structure

The keyword table, shown in Figure IV-5, is the main driving data structure behind the translation module. It contains the information necessary to identify whether or not any particular string is a keyword. If the string is a keyword its DRIF representation is contained in the table thus making the translation process table driven. Also, each TEL operator is assigned a precedence value which will be used in parsing an expression. If the precedence value is  $\emptyset$ , the keyword is not an operator. Finally, the table assigns an index value to each keyword. This index will be used to select a semantic routine to handle the keyword. The most significant hexadecimal digit of the index represents the class of keywords which have been assigned as follows:

<u>DIGIT</u>	<u>CLASS</u>	<u>KEYWORDS</u>
1	Specials	(, ), ;, EOF
2	Unary Logical Operators	NOT
3	Binary Logical Operators	AND, OR, WITH
4	Binary Geographic Comparisons	ALONG, INSIDE, OUTSIDE
5	Binary Relational Comparisons	LT, LE, EQ, GE, GT, NE, CONTAINS
6	Geographic Area Delimiters	CIRCLE, ROUTE, POLYGON
7	Quantifiers	EVERY, NO, SOME
8	Phrase Identifiers	FILE, HOST, RETRIEVE, SHOW

The second digit of the index identifies the specific keyword within this class. As may be seen from the keyword table there is plenty of room for expansion of the number of classes as well as the addition of new keywords within any of the existing classes.

The current memory requirements for the table are 308 bytes for 28 entries. Each new entry will add 11 bytes to the table length. Since the table is sorted alphabetically



TEL KEYWORD	DRIF EQUIVALENT	INDEX	PRECEDENCE
(	0	10	10
)	0	11	10
;	0	12	18
ALONG	18	40	48
AND	1	30	30
CIRCLE	20	60	00
CONTAINS	16	50	48
EOF	0	13	00
EQ	14	51	48
EVERY	30	70	00
FILE	0	80	00
GE	11	52	48
GT	10	53	48
HOST	0	81	00
INSIDE	18	41	48
LE	13	54	48
LT	12	55	48
NE	16	56	48
NO	31	71	00
NOT	3	20	40
OR	2	31	28
OUTSIDE	19	42	48
POLYGON	22	61	00
RESPONSE	0	84	00
RETRIEVE	1	32	20
ROUTE	21	62	00
SHOW	2	83	00
SOME	0	72	00
WITH	4	32	28

8 bytes

1 byte each

NOTE: All character strings are left justified with trailing blanks in a 8 byte field and all numbers are in hexadecimal notation.

Figure IV-5. Keyword Table



by TEL keyword, additions to the table will be made by merging the current table with a list of sorted additions. Deletions will be made by deleting the table entry and compressing the keyword table. The DRIF Equivalent field, the index value, and the precedence value must be assigned to any new additions by someone knowledgeable in the language TEL as well as in DRIF.

#### (2) Operator Stack Data Structure

The Operator Stack is shown in Figure IV-6. Each entry consists of a single pointer to the appropriate entry in the keyword table. This means that several entries in the operator stack may point to a single entry in the keyword table. The three pointers associated with the stack are the current entry pointer and the two pointers delimiting the stack boundaries. The operator stack will be used to remember operators which have already been scanned from the TEL query but have not yet been added to the DRIF. One hundred bytes or 50 entries should be more than enough for the operator stack.

#### (3) Operand Stack Data Structure

The Operand Stack contains all those operands not yet matched with their respective operators. Each time an operator is popped from the operator stack the appropriate number of operands are popped from the operand stack and a resultant operand is pushed back on. All resultant operands will be of type boolean and are denoted by a  $\emptyset$  entry in the stack. All other operands on the stack are represented by a pointer to their location in the DRIF. Thus, each entry in the operand stack consists of 2 bytes. A reasonable size for this stack is 50 entries or 100 bytes. The operand stack is shown in Figure IV-7.

# OPERATOR STACK

## KEYWORD TABLE

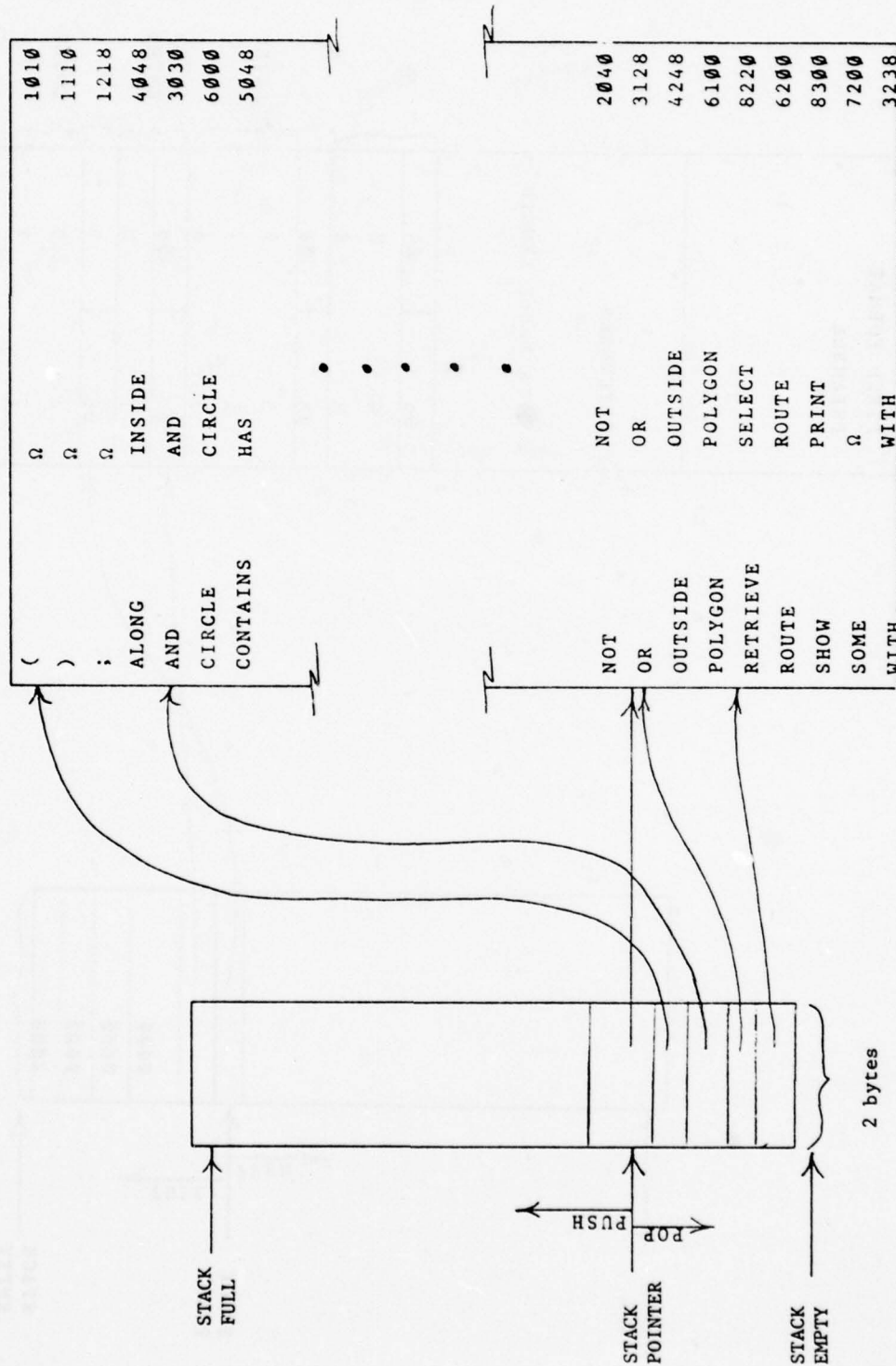


Figure IV-6. Operator Stack

# OPERAND STACK

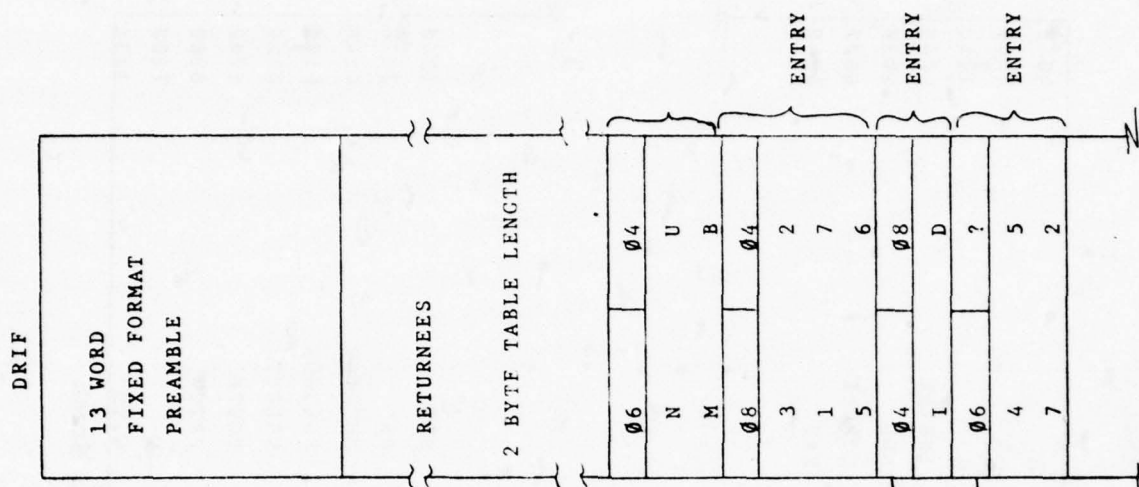
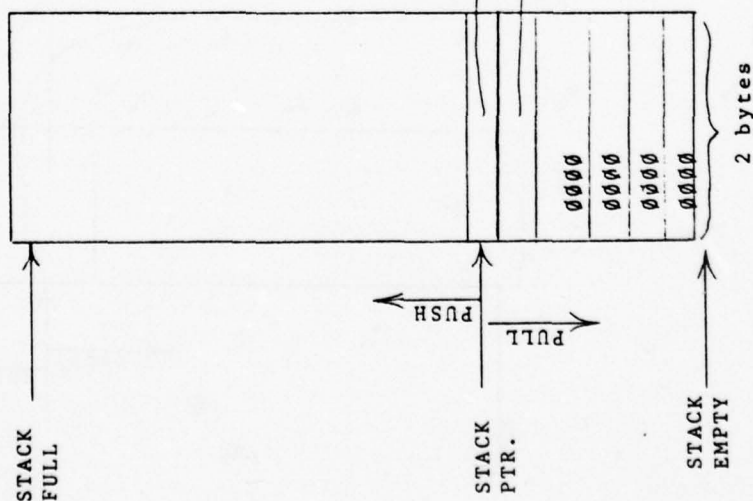


Figure IV-7. Operand Stack

## APPENDIX A

### DEVELOPMENT BACKGROUND

#### 1. TOSS BACKGROUND

Over the past four years, Terminal Oriented Support System (TOSS) concepts and implementation methodologies have undergone research and development by INCO, INC., of McLean, Virginia, for the Rome Air Development Center (RADC). The objective is to provide intelligence analysts with transparent access to computer files, remote data bases and other analysts world-wide through a computer-based communications network. TOSS development has proceeded along lines which seek to extend the concepts of transparency and distributed processing to all relevant aspects of the intelligence ADP environment.

The approach taken by INCO in implementing these concepts has been to build upon existing technology, such as that used in the NMIC Modernization project. Briefly, TOSS is a multifaceted information management and handling system which aids in integrating distributed intelligence resources in a cohesive, world-wide network. It is a hardware/software system employing minicomputers as stand-alone, terminal-oriented processors connected to local host computers to form nodes, which are in turn interconnected by communication links to form a widespread network. The principal components of TOSS include TOSS Exchange Center (TEC) providing network communications control for bulk data and conversational message traffic; Terminal Independent Support System (TISS) providing the capability for on-line analyst interaction through transparency of communication protocols; TOSS Information Management System (TIMS), a data management system providing on-line capabilities for network file access and development and query of hierarchical files; and finally, the Terminal Transparent Display Language (TTDL) facilitating the development of applications programs and stressing the concepts of terminal transparency.

#### 2. TIIN DEVELOPMENT

The TIIN development effort seeks to extend the transparency and distributed processing concepts of TOSS to include query language transparency and data base transparency. TIIN will be based on the hardware/software environment of TOSS with its access to a widespread network of data base management systems with different query language, different technologies for file structure, different conventions for encoding data.

The eventual aim of the TIIN is to allow the user/analyst to converse with the network without regard for file structure and host dependent considerations. The user should feel as though he is accessing information available to him at this local node when, in fact, he is accessing remote data bases, perhaps world-wide.



The TIIN development follows an integrated, step-by-step approach consisting of successive development stages. Each of the stages of TIIN development provides a discrete advance in user capabilities. Advances in system capability require corresponding technological advances. The development of a technology base for TIIN is planned to be systematic and cumulative; most technological features are applicable to all TIIN development stages and are enhanced at each level.

The user capabilities added at each TIIN development stage are described below:

a. Transparent Access to a Local Data Base

The analyst may obtain a description of entry to and use of a local minicomputer data base without specific knowledge of the data base management system.

b. Transparent Access to Distributed Homogeneous DBMS

Local and remote data bases having the same data management system appear as a single data base from the viewpoint of analyst accessibility. Capabilities such as stored queries and distributed standing queries are to be integrated into the network.

c. Transparent Access to Distributed Heterogeneous DBMS

Data base resources available to the analyst are extended to include data bases accessed by data management systems other than a single (common) DBMS, but which are still within the minicomputer network.

d. Integration of Host Computer and Foreign Networks

The network available to the analyst is extended to include data bases in host systems and foreign networks which are connected to the minicomputer network via its nodes. The concept of a network user is extended to include analysts not connected to one of the host systems.

e. Integration of Work Stations Having Intelligent Terminals

Network capabilities are fully integrated with specially designed features and capabilities afforded by an intelligent terminal implemented as an analyst work station in the intelligence community.

3. GUIDING CONCEPTS

The development of the Transparent Integrated Intelligence Network has proceeded in accord with several major principles. They include transparency, data base integration, data sharing, and distributed processing. Each of these ideas will be dealt with in this subsection.

a. Transparency

This concept implies that detailed system knowledge is not required by the user to access and operate network elements. In a transparent environment, analysts can access information from unfamiliar data bases using procedures with which they are familiar. Query translation, data translation, and reconciliation are provided by special software modules, relieving analysts and network elements from requirements for mutual familiarity.

The notion of transparency, as applied to an information processing system, refers to those system characteristics which allow a user to ignore, indeed be unaware of, complexities and diversities which are not relevant to his purpose. For example, problem oriented languages such as COBOL and FORTRAN make machine language transparent to their users.

The concept of transparency as used in the TIIN effort is best communicated in terms of levels of transparency.

(1) No Transparency

Without transparency the analyst must use separate terminals independently connected to each external system. He must use the precise system access procedures as well as that query language associated with the data base he is accessing (the native language of the data base).

(2) Communications Transparency

At this level, the analyst would be able to retrieve data from external systems at his own analyst station, that is, communications and line protocols would be made transparent to the analyst, as well as logon procedures to the external systems. The analyst would still be required to use the data base access procedures (query languages) of the various external systems in order to retrieve data.

(3) Query Language Transparency

At this level, the analyst is able to retrieve data from diverse data bases and data files using a single query procedure from his own terminal. He does not have to know the query language on the particular external data base. There are, however, demands made on the analyst's time and knowledge inasmuch as he must be aware of the dispersion, structure and conventions for data bases being accessed.

#### (4) Data Base Transparency

At this level, dispersion of data resources among the various files and separate data bases is transparent to the analyst. Data is retrieved using one data access procedure from the analyst's terminal; computer software is used to separate the single query into separate queries to be directed at various data bases and files containing the requested information. In addition, the disparity among similar data element names and coded data field values is resolved.

##### b. Integrated Data Base

The intelligence community shares data requirements. The accuracy and completeness of an intelligence analysis presupposes that all relevant data be available to the analyst.

The concept of an integrated DBMS is concerned with providing all users with access to all data bases on a network. The concept presupposes the existence of a common data access language which is implemented system-wide. Integration may be achieved in many different ways: merge all data bases, provide a common DBMS with distributed data bases, or provide query language transparency and data base transparency for existing systems.

##### c. Delegated Production

The concept of delegated production seeks to minimize data maintenance costs by designating specific sites with non-overlapping responsibilities for data collection. A natural corollary of designating specialized data production centers is that analysts must be able to access and cross correlate the data from the non-overlapping data bases. The volume of data updates and the size of the data bases involved make it non-cost-effective to maintain copies of the data bases at the data analysis sites separated from the data collection sites.

##### d. Distributed Processing

The concept of distributed processing seeks to allocate computer resources on a network basis so that response time and overall network utilization are optimized. Balanced availability of computer resources permits the community's computational requirements to be satisfied even when some processors become temporarily unavailable to the network.

#### 4. THE ANALYST

There are two distinct categories into which user analysts



may be classified. The first comprises those who have a non-computer background (perhaps in area studies, languages, the liberal arts) who use a technician to interact with the system. The other class of user analysts has greater knowledge of ADP and can be expected to exploit the system to a greater degree than the first group. This latter group includes data specialists, file sponsors, and data base administrators.

Though both groups may benefit from the development of transparency oriented processing, it is probably the non-computer oriented analysts who may benefit most from increased user-orientation of data access. To enhance productivity and to utilize the analyst's problem-oriented training, the analyst cannot be unduly concerned with the mechanics of data retrieval or with the arbitrary differences between query languages, file structures, and data codes on a distributed network of diverse intelligence resources.

a. Analyst's Task

The intelligence analyst is responsible for assessing information related to critical and complex situations which have far-ranging political and military significance both nationally and internationally. The analyst frequently must work under considerable time pressure. To support his analysis and judgemental capabilities, a large volume of diverse information exists in multiple and uncoordinated sources. Analysts have traditionally developed highly personalized and often very sophisticated manual techniques for accessing information based on experience and knowledge of sources. While the analyst may resort to rather sophisticated "shoe-box" techniques for storage of retrieved data, the main concern is not data storage or retrieval, but analysis.

The analyst's basic mission is to monitor and interpret intelligence information and report it in a timely and effective manner for further assessment at decision-making levels. In fulfilling this mission, he performs the following activities.

- o Monitoring messages, indicators, and events
- o Establishing and maintaining data files
- o Seeking information from other analysts and data files
- o Preparing finished intelligence reports
- o Joint assessment of current intelligence situations
- o Verifying and enhancing information through cross-correlation.



These activities are conducted within the context of three functional levels related to the urgency of the situation at hand. For the indications and warning (I&W) analyst these are:

- o Watch Function - monitoring indicators and assessing current events
- o Current Intelligence - actively tracking significant events and crisis situations
- o In-depth Analysis - developing background information and analyzing information with long-term implications.

These functions are keyed to the operation of command centers and the formulation of precise military decisions based on the best possible intelligence information. Within this context, the analyst's objectives are timely and accurate assessment and subsequent reporting of complete intelligence data. To accomplish these basic, but difficult, objectives the analyst has very real requirements which can be met through data processing support.

b. Data Analysis Requirements

There is a strong and continuing need for automated support to provide the analyst with ready access to data. A large measure of automatic data processing support is being developed using facilities already in place. However, these facilities are geographically dispersed and reside on independent hardware and software systems.

A major need is on-line access to data bases which are not part of the analyst's immediate system. To be useful to the analyst, ADP support must make minimal demand on him to learn access techniques. The ideal ADP support should unburden the analyst from an already substantial data processing workload. This requires the implementation of interactive, integrated networking concepts with sufficient translating and routing software to permit the analyst to access external systems in his own familiar terms. The provision of simplified, real-time access to remote data bases would greatly aid the analyst in cross-correlation, data assessment and reporting functions.

The volume and variety of intelligence data, and the ways in which it may be stored and handled in the intelligence community are great. Merely to monitor the vast array of information provided by sensors and other data collection devices within the analyst's area of specialization can be an immense job; the cross-correlation of data is staggering. Requirements for automatic data processing support clearly exist both with regard to the acquisition of data and its later manipulation for analytic purposes.

Data processing needs are critical to the nature of the indications and warning analyst's mission, functions, and activities. Automation can support not only the monitoring activities in which the analyst is frequently engaged, but will greatly assist the analyst's performance in crisis situations.

Analyst data needs amenable to the application of ADP technology generally focus on the access and manipulation of data. Both access and manipulation have several facets. Access requirements are broad and specific: the analyst has a great interest in a wide variety of sources and indicators, as well as specific information requirements. Since the analyst desires to keep abreast of developing situations, single elements of intelligence are sometimes critical; timeliness, validity, and reliability are also very important. The data manipulation needs of the analyst focus on the requirement to rapidly process and format large volumes of diverse information. By way of illustration, some access and manipulation needs are: immediate updating of key indicators, corroboration of hypotheses with initial intelligence multi-source cross-correlation support, program generation for formatting unfamiliar data packages, real-time requests for specific information, and queries to experts outside of the analyst's field of specialization.

## APPENDIX B

### DESCRIPTION OF THE TIIN SYSTEM

#### 1. INTRODUCTION

The purpose of this appendix is to provide a concise description of the function of the components of the Transparent Integrated Intelligence Network (TIIN). For development purposes the components are grouped into five subsystems to be specified and developed in the following order:

TIIN/TILF	data description and directory management
TIIN/QIP	host selection and host query generation
TIIN/RN	report normalization and report formatting
TIIN/TAP	analyst aids processor and query normalization
TIIN/QDNC	query distribution and network interface.

These subsystems and their components are shown in Figure B-1. This figure will be referenced throughout this appendix. The figure illustrates the distinction between user-dependent components and host-dependent components. It is assumed that host dependent components exist only at the host node, and user-dependent components exist only at the user node, since this minimizes the number of copies of the host dependent components in the network. One alternative is to place all components at the user node, which would reduce bottlenecks at the host node as a tradeoff for duplicating all host-dependent components at every user node.

Briefly, the key to understanding the TIIN system is the concept of normalization, which refers to the use of internal standards to serve in the absence of standards between the interconnected host computers. Three areas of normalization can be considered to be the province of the TIIN system specifically, query normalization, report normalization, and data structure normalization. Other areas of normalization, such as host access protocols, packet format, and terminal interface, are the province of the network software/hardware system. It is assumed that TIIN will operate on the PDP-11/45 under RSX-11D, using the software components of the Standard Software Base (SSB) and the Terminal Transparent Display Language (TTDL). The network interface will be based on WICS Common Format (WFC) or its successor.

Normalization allows the user to have many of the advantages of standardization without causing the hosts to entail the disadvantages of standardization. Through normalization the enormous range of differences between the host systems becomes transparent to the user.



a. Data Structure Normalization

The TIIN internal convention for data structure presupposes that the host data base consists of a collection of interrelated hierarchical files. A record in a hierarchical file consists of a collection of hierarchical dependent segment, each segment consisting of a fixed number of variable length single valued fields. Each segment except the root segment has a single parent segment, and each segment type may occur a variable number of times in its dependency relation with its parent segment. There is one root segment per record, and all segments in the record are either directly or transitively dependent on the root segment.

The normalized data structure is stored in the Network Access Directory (NAD) by the Data Base Administrator (DBA). The NAD contains all the data descriptions needed by the TIIN components. The hierarchical relation between segments is represented by indicating the identifier of the parent segment for each segment.

b. Query Normalization

The TIIN internal convention for queries, called Query Normal Format (QNF), is essentially reverse Polish notation with all names and values explicitly represented and flagged to indicate whether they are in the user, network, or host frame of reference. At the user node the QNF is called Data Request Intermediate Format (QTF) to denote the internal representation used for performing algebraic transformations to convert the query into the host query language. The QTF and the associated transformations are substantially different for each host.

c. Report Normalization

The TIIN internal convention for reports, called Response Normal Format (RNF), consists of a file descriptor followed by a sequence of records. The response file is completely self-descriptive and is thus completely independent of the files described in the NAD. The RNF version of the report is suitable for further processing: it may be formatted as a user report, or input to an applications program, or stored as a file in the analyst data base.

2. TERMINAL ACCESS PROCEDURES (TAP)

Components of the TAP subsystem appear in the upper middle and left section of Figure B-1. The subsystem includes the Analyst Aids Processor, and the Query Language Processor.



The TAP subsystem is designed to provide a convenient and natural user interface method. The functions of TAP are 1) to describe network data resources to a user/analyst, 2) to support the user in the construction of legal queries, and 3) to allow for rapid, interactive dialogue with the network.

a. Query Language Processor

The Query Language Processor will provide the analyst the capability of expressing a query via a simple formal language. The module will check the syntactic correctness of the query and will produce the normalized version (DRIF) of the query for the Query Intermediate Processor (QIP) at the user node.

b. Analyst Aids Processor

The descriptive information which is maintained in the Network Access Directory for use by TIIN components is also useful to the analyst in identifying the precise data requirements. The Analyst Aids Processor displays the contents of the NAD in a user-oriented format.

NAD information is displayed by subject category with prose descriptions of files related to subjects the analyst selects, prose descriptions of data elements belonging to selected files, and properties and attributes of selected elements.

3. TRANSPARENT INTELLIGENCE LANGUAGE FACILITY (TILF)

The components of the TILF subsystem appear in the upper part of Figure B-1. This subsystem is devoted to describing the contents of host data bases and keeping these descriptions up-to-date for real-time access from any node in the network. The subsystem includes the User Data Description Processor, the Host Data Description Processor, and the Network Access Directory.

a. Network Access Directory (NAD)

The Network Access Directory (NAD) is the repository for all information required by the TIIN system about the contents of network data bases. Individual versions of the NAD will be created and maintained at each network node, and either all or part of each local NAD must be able to be easily transmitted to other network nodes on request. This feature will enable each node to periodically gather and consolidate information from its own and other NADs, consolidate the information, and use it to locate data throughout the network.

The largest segment of a local NAD will be a file, each entry of which contains descriptive information about a local data element, a network standard element (which may or may not have a corresponding local element), or a local-usage name for a network data element. Other locally-maintained NAD files will contain descriptive subject and routing information for the parent files and parent data bases of local elements and legal values or ranges of values for local elements.

Data Element entries must be accessible via two modes of user access. Normally, specific data elements will be "found" when an analyst expresses an interest in a particular subject; however, from frequent use of the network system, the analyst may choose to refer by name to elements, thus bypassing perusal of subject categories. The information about a data element includes:

- o Normalized name
- o Element type (string or numeric)
- o Length and format information
- o Identifier of parent file
- o Identifier of parent data base
- o Value translation algorithm or table.

b. User Data Description Processor

The User Data Description Processor is one of the User modules in the TIIN. Unlike the Query Language Processor and the Analyst Aids Processor, this processor is password protected. It is intended for use only by the User Data Base Administrator (DBA), not by TIIN analysts.

The User Data Description Processor creates and updates the User NAD. Its function is to provide a means for the User DBA to define that portion of the network data which is available to the local analysts.

c. Host Data Description Processor

The Host Data Description Processor is one of the Host TIIN modules. The Host Data Description Processor is privileged and password protected. It is intended to be used by a Host DBA only.

The Host Data Description Processor creates and updates the Host NAD. Its function is to provide a means for the Host DBA to describe to the network those files and those elements which are potentially accessible to all User nodes. In addition, it provides a means for the Host DBA to perform updates whenever necessary and to display information contained in the NAD.

Both the User Data Description Processor and the Host Data Description Processor run in the interactive mode with their respective DBA. Neither DBA is required to know the exact structure of the NAD. Rather, what is important is that they understand the structure of the data bases which they are defining.

#### 4. QUERY INTERMEDIATE PROCESSOR (QIP)

The QIP subsystem appears in the center of Figure B-1. This subsystem deals with host selection, element name translation, and query translation. The user-dependent portion of QIP (User QIP) is called the Translation and Data Base Selection module. The host dependent portion of QIP (Host QIP) is called the Host Query Language Generator.

The functions of the User QIP are 1) selection of host data bases and files containing the elements requested by the user, as indicated in the DRIF, and 2) translation of element names and values from the user to the normal TIIN frame of reference.

The host QIP translates the user query from the TIIN normal format (QNF) into the host DBMS query language.

The sequence of actions through which the query will be processed includes:

- o Identification of those files which contain the requested data elements
- o Substitution of network-standard element names for synonymous user element names
- o Creation of equivalent versions of a query for addressing different files which may contain similar data
- o Creation of sequences of queries when several files must be searched in sequence to retrieve data, or when several different search verbs are necessary to process all the search criteria



- o Reconciliation of distinctive functions in the QNF not provided for in the host language
  - o Translation of network standard element names into element names in the host frame-of-reference
  - o Generation of the host version of the user query.
5. QUERY DISTRIBUTION EXECUTIVE/NETWORK COMMUNICATIONS INTERFACE (QDNC)

The QDNC subsystem appears in the middle of Figure B-1. This subsystem deals with 1) routing of queries to data resources, both local and external, 2) tracking and logging of incoming and outgoing network communications, and 3) the initiation of transmission of data from local to external nodes.

a. Query Distribution Executive

This module tracks and routes queries and responses. For incoming queries, it makes an entry in the incoming query queue with status of "awaiting processing." The Distribution Executive polls the outgoing query queue for entries with "response received" status. Upon finding one, the Distribution Executive notifies the report and display function passing the message sequence number of the response file.

b. Network Communications Interface

The Network Communications Interface initiates transmission of data from the local node to any other network node and receives data into the local node from other network nodes. The data which will pass through the Interface will consist of:

- o Queries being distributed to other network nodes for processing
- o Incoming responses from distributed queries
- o Queries coming into the node from other network nodes
- o Outgoing responses to queries from other nodes
- o Bulletins from one network to all other nodes to announce that a NAD update has taken place at the originating node



- o A request from one network node to a second node for transmission of the second node's NAD segment
- o A NAD segment transmitted from one network node to another.

Incoming and outgoing messages are blocked and routed using the communications modules of TISS. An outgoing "message" (any data transmissions between network nodes), is converted to a WCF (WICS Common Format) header block including the message sequence number (MSN), user identification, node identifier, message type (query, query response, NAD bulletin, etc.) Outgoing message traffic is handled by polling two queues: the outgoing query queue (for queries to be sent), and the incoming query queue (for responses to be sent). Incoming message traffic is also handled, with most message types being passed immediately to other network elements for processing.

#### 6. RESPONSE NORMALIZATION (RN)

The RN subsystem appears in the bottom portion of Figure B-1. The Response Normalizer, in conjunction with the Response Normal Format (RNF), is designed to provide a capability for output standardization in the overall TIIN design. In general, the task of the RN is to provide the capability for converting data to a standard format after it has been retrieved from a host DBMS. Since data which is returned in response to a single query may be from separate files, it is likely to exhibit different characteristics with regard to coding conventions. The RN normalizes responses, converting them to a standard format which enables collation and reconciliation of conflicts and duplications. The RN also facilitates storage of data in the analyst's private file where it would be available for further processing (such as refining the original request, sorting, statistical analysis, graphic display, etc.).



## APPENDIX C

### GLOSSARY

- COINS - The Community On-line Intelligence Network Subsystem provides batch oriented access to intelligence data bases at NSA, DIA, CIA and NMIC.
- DBDL - The Data Base Description Language is a TILF-supported language used to describe the contents and structure of network data bases.
- DBMS - A Data Base Management System manages and accesses a data base and provides responses to queries received from users.
- DIAOLS - The Defense Intelligence Agency On-Line System is a host DBMS on the COINS network.
- DRIF - Data Request Intermediate Format is a data structure used to pass queries from a user (e.g., TAP) to QIP at the user node. The DRIF is query language independent.
- Host (DBMS) - In this report, a host always refers to a data base and its associated DBMS accessible via a communications network.
- Host NAD - A part of the NAD containing information used by the Host QIP at the host node to translate network standard format (QNF) queries into a host query language.
- Host Node - In this report a host node is a network node which provides access to one or more data bases of an integrated intelligence network. The TIIN software at a host node will receive a QNF version of a user query and will perform operations on it to make it acceptable to the host data base management system. The DBMS provides a response to the query, which is sent to the TIIN software at the host node for normalization, preparatory to returning the query result to the user node on the network.
- Host QIP - The Query Intermediate Processor at the host node validates a query given the restrictions imposed by the host query language and data base, and translates the query into the host query language.
- Host RN - Those modules of the RN subsystem dealing with DBMS response processing at the host node. The function of the Host RN is to normalize DBMS responses into a TIIN standard response format, RNF.
- NAD - The Network Access Directory is a distributed data base containing information about intelligence data bases in the network. Information contained in the NAD allows for element name translation, value translation, and host selection during TIIN processing.

AD-A055 387

INCO INC MCLEAN VA  
TERMINAL ACCESS PROCEDURES (TAP).(U)

F/G 9/2

APR 78 K WELLS, M HUBERT, N HAUSER

F30602-77-C-0045

UNCLASSIFIED

INCO/1090-178-FR-3-D(F)

RADC-TR-78-90

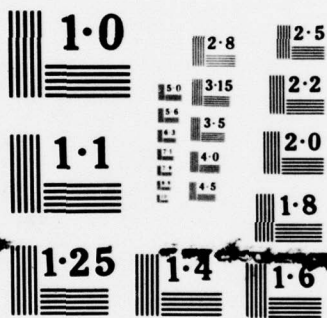
NL

2 OF 2  
ADA  
055387



END  
DATE  
FILMED  
8-78  
DDC





NATIONAL BUREAU OF STANDARDS  
MICROCOPY RESOLUTION TEST CHART

NDC - The Network Data Catalog provides a cross index by subject of the files in the NAD.

Network - A network is a complex consisting of two or more interconnected computers.

NMIC - National Military Intelligence Center, Arlington, Virginia.

Node - A data processing site with its own communications interface to an extended, integrated network. In this report it is assumed that a node will have a PDP-11 with TOSS software, including TIIN with a Network Access Directory.

QDNC - The Query Distribution and Network Communications subsystem is a proposed TIIN subsystem concerned with the distribution of a query into the extended network and the tracking of queries distributed and responses received. It interfaces with the TOSS communications handler (TISS) which handles all message traffic to and from a node.

QIP - The Query Intermediate Processor is the collection of TIIN modules which transform a user data request into a standard query language and data base independent format for transmission to host nodes, and subsequently into a format compatible with a host node's data base management system.

QNF - Query Normal Format is a compact data structure used to transfer queries from the user node to the host node.

QTF - The Query Tree Format is a data structure used by the host QIP at the host node, to allow easy validation and restructuring of the query.

RADC- Rome Air Development Center, Griffiss Air Force Base, Rome, New York.

RN - Response Normalization is a proposed TIIN subsystem which will receive the response to a user query from the host DBMS, convert it into TIIN standard format (RNF) for transmission to the user node, merge all standardized responses resulting from a single user request (DRIF) and present the user with a finished report.

RNF - The Response Normal Format is a TIIN standard data structure, used to transmit a query response over a network.

RADC - Rome Air Development Center, Griffiss Air Force Base, Rome, New York.

RN - The Response Normal Format is a TIIN standard data structure, used to transmit a query response over a network.

RPN - The Reverse Polish Notation is a representation technique in which operand expressions occur in a linear sequence immediately before the associated operator.

- SSB - The Standard Software Base is a collective phrase encompassing TOSS-related and other software which provides a baseline for Air Force network applications.
- TAP - The Terminal Access Procedures is a TIIN subsystem which allows a user to make a data request and to gain information about data base content and structure. TAP includes a language translator to produce the DRIF version of the user query to interface with the user QIP.
- TEC - TOSS Exchange Center provides network communications control for build data and conversational message traffic.
- TEL - The Transparency Examples Language is a hypothetical query language used to illustrate transparency features relative to a host query language.
- TIIN - Transparent Integrated Intelligence Network.
- TILE - TIPS Interrogation Language.
- TILF - The Transparent Intelligence Language Facility is a TIIN subsystem concerned with creation and maintenance of the Network Access Directory (NAD).
- TIPS - Technical Information Processing System, a DBMS at NSA.
- TOSS - The Terminal Oriented Support System is a multi-faceted information management and handling system which aids in integrating distributed intelligence resources in a cohesive, worldwide network. It is a hardware/software system employing minicomputers as stand-alone, terminal-oriented processors connected to local host computers to form nodes, which are in turn interconnected by communication links to form a widespread network. Components of TOSS include TEC, TIMS, TISS, TTDL.
- TTDL - Temrinal Transparent Display Language is a transparent terminal handling subsystem of TOSS.
- User NAD - A part of the NAD containing information used by the User QIP at the user node to translate user element names and values into TIIN standard element names and values.
- User Node - The user node is a network node which provides terminal support through which an analyst initiates a query. A user node may have an associated DBMS. The TIIN software at the user node is where the user query is initially processed for distribution to host nodes.

User QIP - The Query Intermediate Processor at the user node consists of a single module, the QNF Generator. The functions of the User QIP include element name translation, data value translation, and host selection.

User RN - Those modules of the RN subsystem dealing with DBMS response processing at the user node. The function of the User RN is to merge all standardized responses resulting from a single user request (DRIF) and to generate a report in a format requested by the user.

WICS - Worldwide Intelligence Communications System.

WCF - WICS Common Format.



## APPENDIX D

### IMPLEMENTATION LANGUAGE

#### 1. PROGRAMMING LANGUAGE CHOICE

In order to support the development of the Analyst Aids Processor the User Data Description Processor, the Host Data Description Processor, and the Query Language Processor under the TAP contract, a development language must be chosen. The language should be high level to speed the development process and should be capable of running on the PDP-11/45. Our choice to meet these criteria is the "C" language. It was written at Bell Telephone Laboratories to run under the UNIX Operating System on the PDP-11/45. A version which will run under the RSX-11 Operating System has been released by Yourdon, Inc. This version is useable in our application.

#### 2. SELECTION CRITERIA

The choice of "C" was based primarily on language surveys done by two other groups at INCO. The first was done as part of the IDHSC-II project and is available as Project Memo 1094/16. It included evaluations of PASCAL, ULP, BLISS-11, FORTRAN IV, ALGOL, SIMPL-T, PL/1 and SP/k as well as the C language. In this survey C met the standards for all evaluation criteria except for RSX-11 compatibility. This deficiency should be eliminated with the release of the Yourdon Compiler. The second language survey was performed under the IDPN contract and it is documented in Technical Memo 1095/07. The languages mentioned in this survey include "C", PASCAL, FORTRAN IV, COBOL, RATFOR, BLISS-11, PL/1, SIMPL-T, MODULA and several assembly language macro packages. The choice of "C" as a development tool is based on the evaluation of the Bell Laboratories "C" compiler which runs under UNIX. The language differences between this version of "C" and the RSX-11 compatible Yourdon version is minimal. The selection criteria includes the following points.

- o The language has been implemented on a PDP-11 mini-computer and is available under RSX-11
- o The code produced by the compiler is efficient both in terms of core requirements and speed of execution.
- o The language allows access to the hardware registers
- o The language has a good set of control structures thus facilitating structured programming techniques
- o The language supports character string operations
- o The language allows the programmer control over data structures

- o The output of the "C" compiler allows for modularization of a system. Any system may be broken into separately compilable modules.
- o Minimal run time support is needed for programs coded in "C"
- o Conditional compilation is available
- o "C" programs will run on the PDP-11 under UNIX. Thus there will be no need to rewrite all the software if programs are moved from an RSX-11 environment to a UNIX environment.
- o "C" has been implemented on the Honeywell 6000, the IBM 360, the IBM 370, and the PDP-10 computer systems. This allows portability of software to other machines as well as other operating systems.

### 3. PLAN OF ATTACK

The "C" Language has been selected as the language in which to implement TIIN designs. A big advantage of this choice is the portability of programs written in "C" from the RSX-11 environment to the UNIX Operating System environment. Given the time frame of the TAP contract, however, it was necessary to perform our coding in a language already available. This temporary implementation language is RATFOR. It provides both the advantages of structured programming and the availability of all the system support software written for FORTRAN. The second feature exists because the RATFOR language is essentially an extension of FORTRAN implemented by means of a RATFOR "preprocessor" which converts RATFOR source code to FORTRAN source code. A primary example of support software valuable to TIIN is the Terminal Transparent Display Language (TTDL). FORTRAN interfaces to TTDL already exist and may be used in RATFOR programs. Another advantage of our current use of RATFOR is its similarity to the "C" Language.

#### a. Compatibility with C

The RATFOR language bears a high resemblance to the C language. Both support IF, ELSE, FOR, WHILE and block structure constructs along with the basic arithmetic, relational and logical operators. See Figure D-1. The resemblance between C and RATFOR is more appearance than reality. In an experiment a program written in C was hand-converted to RATFOR. Over 80% of the lines of code had to be revised. This incompatibility factor occurred even though the C programmer had avoided intrinsic C features such as the PL/I data structures and pointer processing. Much of the incompatibility resulted from the difference in comment conventions, array subscripting, and auto-increment oper-

#### Features in RATFOR Extension

/\*comment\*/  
negation operators (!,!=)  
DEFINE (AND,&)  
DEFINE (OR,|)  
array [expr] 0...N-1 right-to-left  
DO; and END; for block structure

#### Features in both C and RATFOR

IF (cond) stmt  
ELSE stmt  
FOR (initial; cond; iterate) stmt  
WHILE (cond) stmt  
\$ (and\$) for block structure  
arithmetic operators (+,-,\*,/)  
relational operators (<,>, =, >=, <=)  
logical operators (&,|)

#### Features in C, not in RATFOR Extended

auto-increment (++)  
auto-decrement (--)  
EXTERNAL declaration  
compact I/O routines  
parameters-by-value  
reentrant code only  
recursive calls  
full PL/I data structures  
pointer operators (->,\*+,+,-)  
sub-arrays and sub-structure  
subroutine (args);  
special operators (%,&, <<, >>, etc.)  
logical operators (&&, ||, ?, :)  
assignment operators (=, +=, -=, etc.)  
DO stmt WHILE (cond);  
SWITCH ... CASE ...  
string (") vs. character (') distinction

#### Features in RATFOR not in C

#comment  
array (expr) 1...N left-to-right  
BLOCK DATA  
COMMON declaration  
DATA  
FORMAT driven I/O  
READ  
WRITE  
parameters-by-reference  
non-reentrant code  
non-recursive calls  
negation operators (¬, ¬=)  
\*\* exponentiation operator  
one "=" per stmt  
CALL sub (args)  
DO var=, init, terms stmt

C

RATFOR  
(Rational  
FORTRAN)

Figure D-1. Summary of Difference between C and RATFOR



ators. For example, RATFOR loops run from 1 to N while C loops run from 0 to N-1, so even though the FOR construct is identical between C and RATFOR, all the FOR statements had to be revised.

b. RATFOR Extended

The RATFOR preprocessor has been extended to process C-style comments and C-style array subscripts. The purpose of these extensions is to provide a vehicle for writing and debugging C-style programs. With these extensions it is practical to write programs which are over 80% compatible with C, i.e., less than 20% of the lines of the code will require modification to use a C compiler.

Figure D-2 illustrates the differences between programming in RATFOR versus RATFOR Extended. Note that RATFOR-style subscripts use parentheses, run from 1 to N, and assumes arrays are stored column-major (first subscript varies fastest), whereas C-style subscripts use brackets, run from 0 to N-1, and assume arrays are stored row-major (last subscript varies fastest).



```

1.  DEFINE(NI,100)
2.  DEFINE(NJ,4)
3.  SUBROUTINE EXAMPLE
4.  INTEGER B(NJ)
5.  INTEGER A(NI, NJ)
6.  DATA B/4*222/
7.  DATA A /NI*1, NI*2, NI*3, NI*4/
8.  * ARRAY INITIALIZED SO A(I,J) = I
9.
10. FOR (I=1; I<=NI; I=I+1)
11.   FOR (J=1; J<=NJ; J=J+1)
12.     A(I,J) = A(I,J) + B(J)
13.   END
14. END

```

Figure D-2a. Programming Example Written in RATFOR

```

1.  SUBROUTINEEXAMPLE
2.  INTEGERB(4)
3.  INTEGERA(100,4)
4.  DATA B/4*222/
5.  DATA A/100*1,100*2,100*3,100*4/
6.  CONTINUE
7.  I=1
8.  23000 IF(.NOT.(I.LE.100))GOTO 23002
9.  CONTINUE
10. J=1
11. 23001 IF(.NOT.(J.LE.4))GOTO 23003
12. A(I,J)=A(I,J)+B(J)
13. 23004 J=J+1
14. GOTO 23003
15. 23005 CONTINUE
16. 23006 I=I+1
17. GOTO 23000
18. 23002 CONTINUE
19. END

```

Figure D-2b. Equivalent FORTRAN Code Produced from Above  
(Figure D-2a) by Preprocessor.

```

1.  DEFINE(NI,100)
2.  DEFINE(NJ,4)
3.  SUBROUTINE EXAMPLE/
4.  DO/
5.  INTEGER B(NJ)
6.  INTEGER A(NJ)(NI)
7.  DATA B /NJ*222/
8.  DATA A /NI*1, NI*2, NI*3, NI*4/
9.  /* ARRAY INITIALIZED SO A(J)(I) = I+1 */
10.
11.  FOR (I=0; I<NI; I=I+1)
12.  FOR (J=0; J<NJ; J=J+1)
13.      A(J)(I) = A(J)(I) + B(J);
14.  END/

```

Figure D-2c. C-style Programming Example Written in RATFOR Extended.

```

1.  SUBROUTINEEXAMPLE
2.  INTEGERB(4)
3.  INTEGERA(100,4)
4.  DATAB/4*222/
5.  DATAA/100*1,100*2,100*3,100*4/
6.  CONTINUE
7.  I=0
8.  23000 IF(.NOT.(I.LT.100))GOTO 23002
9.  CONTINUE
10.  J=0
11.  23003 IF(.NOT.(J.LT.4))GOTO 23005
12.  A(I+1,J+1)=A(I+1,J+1)+B(J+1)
13.  23004 J=J+1
14.  GOTO 23003
15.  23005 CONTINUE
16.  23006 I=I+1
17.  GOTO 23000
18.  23002 CONTINUE

```

Figure D-2d. Equivalent FORTRAN Code Produced from Above.  
(Figure D-2c) by Preprocessor.

## APPENDIX E

### DBMS INTERFACE

#### 1. REQUIREMENT

The TIIN system in general has a requirement for most of the functions provided by a general Data Base Management System (DBMS). Services such as internal file maintenance, sorting and merging records, generating reports, etc., are all required by some module of TIIN. TIIN/TAP, in particular, has a need for a set of data management services to be used in the construction and maintenance of those files which compose both the Host and User NADs. These files all are hierarchically structured. Data elements are grouped into segments and segments are interrelated as nodes within an n-ary tree. The root node of the tree corresponds to a level 1 segment. Each successive level in the tree represents an additional level of segment nesting and thus an additional level of hierarchy in the file. The first element in the level 2 segment is the record key. The first element in any other segment is the key to location of that particular segment. Elements are never repeated within a segment.

#### 2. DEVELOPMENT STRATEGY

The TAP effort requires the use of a system for construction and maintenance of a series of files. As of now, no particular DBMS has been chosen to meet the requirements of TIIN. The strategy, therefore, has been to define a set of general services for the location of and update of hierarchical files. These services have been used in coding the Analyst Aids Processor, the User Data Description Processor, and the Host Data Description Processor. These services may then be coded as subroutine calls to any selected data base management system. The services used are all concerned with reading and writing records from hierarchical files. Other services such as report generation and file sorting and searching have not been required by any of the TAP modules.

#### 3. SUBROUTINE DEFINITIONS

Each of the routines described below is used in at least one of the TAP processors. The routines are all designed to manipulate hierarchical files. The detailed description of the TIIN format for all hierarchical files may be found in the TIIN/RN Final Technical Report published in December 1977. Below is a description of the function of each of the currently-defined file operators. All character arguments used are left justified with trailing blanks in their respective fields.



a. Open File

This routine is used to open a file for access. The user program specifies whether it is requesting read only access or read/write access. Before a file may be accessed by a program it must be opened. The routine is invoked as follows:

CALL OPEN (file name, file access, status)

file name: six character name of file to be opened

file access: 0 - read access request  
1 - read/write access request

status: an integer value returned by the routine

0 - successful operation (file opened)  
3 - file unavailable  
5 - file not found

b. Close File

This routine is used to close a file. The close operation is performed when the user program no longer requires access to it. The routine is invoked as follows:

CALL CLOSE (file name, status)

file name: six character name of file to be closed

status: an integer value returned by the routine indicating the result of the routine's operation  
0 - successful operation (file closed)  
1 - file not opened

c. Create File

This routine is used to add a new file to the data base. It checks to see if the file already exists and if it does not, the file is created. The routine is invoked as follows:

CALL CREATE (file name, file structure, status)

file name: six character name of file to be created



file structure: address of a table defining the structure of the file. The table consists of a count of the number of entries in the table and the entries. Each entry consists of a pair of integer values, both in the range of 0 to 255. The first value is the segment identifier and the second value is the identifier of the parent segment. Thus, if N is the number of different segments in the file, the file structure table will contain  $2N + 1$  integer values.

status: an integer value returned by the routine specifying the result of the operation.

- 0 - successful operation (file created)
- 4 - file already exists
- 12 - illegal structure table
- 13 - no room for file

#### d. Delete File

This routine is called to delete a file from the data base. The routine is invoked as follows:

CALL DELETE (file name, status)

file name: six character name of file to be deleted

status: an integer status value returned by the routine

- 0 - successful operation (file deleted)
- 5 - file not found

#### e. Select Record

This routine is used to locate a record in a file, given the record key. Once the record is located, it becomes the current record for this file and the routine returns to the user program. The routine is invoked as follows:

CALL SELREC (file name, key, status)

file name: six character name of file in which record will be found

key: the value of the first element in the file's level 0 segment. It is used by SELREC to identify the particular record requested. The key value should be unique within the file.

status - indicates the result of the operation. Possible values are:

- 0 - successful operation (record found)
- 1 - file not opened
- 6 - record key not found

f. Select Next Record

This routine is used to select the record immediately following the current record in the file. The selected record then becomes the new current record. If no record has yet been selected from the file, the first record will be selected. If the current record is the last record in the file, an end-of-file indicator is returned in the status word. The method of invoking the routine is as follows:

CALL SELNXR (file name, status)

file name: name of file (6 characters) from which next record is to be selected

status: indicates result of operation. Possible returns are:

- 0 - successful operation (next record found)
- 1 - file not opened
- 9 - end of file

g. Select Segment

This routine is used to locate a segment within the current record of the specified file. The key used in locating the segment is the value of the first element. Once located, the segment becomes the current segment with the given segment identifier. The routine is invoked as follows:

CALL SELSEG (file name, segment ID, key, status)

file name: six character name of file used to identify the proper current record (if several files are open, several current exists - one for each open file)

segment ID: the segment ID of the particular segment type which will be searched (note: besides segment ID, the current of each ancestor segment determines the group of segments to be searched)

**key:** the value of the first element in the segment type specified used to select the proper segment. The key should be unique within this group of segments with the specified ID.

**status:** the status value returned by the routine indicates the result of the operation.

- Ø - successful operation (segment found)
- 1 - file not opened
- 7 - segment key not found
- 8 - segment ID not in file

#### **h. Select Next Segment**

This routine is used to select the next segment with the specified ID. Note that when selecting the next segment, not only must it have a parent segment with the same ID as that of the current segment's parent segment, but it must actually have the identical parent segment. The routine is invoked as follows:

**CALL SELNXS (file name, segment ID, status)**

**file name** - name of file (six characters) from which segment is to be selected.

**segment ID** - the segment ID of the segment type to be searched

**status** - the value returned indicating the result of the operation.

- Ø - successful operation (next segment found)
- 1 - file not opened
- 1Ø - end of segments

#### **i. Insert Record**

Insert a record into the specified file in front of the current record for the file. The routine is invoked as follows:

**CALL INSREC (file name, status)**

**file name** - six character name of file which will have record inserted

**status** - indicates result of operation.

- Ø - successful operation (record inserted)
- 2 - file not opened for write
- 14 - no room in file for record



j. Delete Record

Delete the current record from the specified file. The routine is invoked as follows:

CALL DELETR (file name, status)

file name - six character name of file to have a record deleted

status - indicates result of operation

0 - record deleted

2 - file not opened for write

9 - end of file,

k. Insert Segment

This routine inserts a segment after the current occurrence of that segment. The routine is invoked as follows:

CALL INSSEG (file name, segment id, buffer size, buffer address, status)

file name - the name of the file containing the record into which the segment will be added

segment ID - the identifier of the segment to be inserted. Segment will be inserted after current occurrence of segment.

buffer address - address of buffer containing segment to be inserted.

buffer size - number of bytes in segment

status - indicates result of operation

0 - successful operation (segment inserted)

2 - file not opened

11 - illegal segment ID

15 - no room in record for segment

l. Delete Segment

This routine deletes the current occurrence of the specified segment. The method of invoking the routine is as follows:

CALL DELETS (file name, segment ID, status)

file name - six character name of file to have segment deleted.



segment ID - identifier of the segment whose current occurrence is to be deleted

status - indicates result of operation

- Ø - successful operation (segment deleted)
- 2 - file not open for write
- 11 - illegal segment ID

m. Read Segment

This routine causes a segment to be transferred into a specified user program buffer. The operation causes the current occurrence of the specified segment to be transferred. The routine is invoked as follows:

CALL READSG (file name, segment ID, buffer size, buffer address, status)

file name - six character name of file containing segment to be input into the program buffer.

segment ID - identifies segment whose current occurrence is to be transferred into the program buffer.

buffer size - size of buffer in bytes

buffer address - address of program buffer to receive segment

status - indicates result of operation

- Ø - successful operation (segment read into buffer)
- 1 - file not opened
- 11 - illegal segment ID
- 16 - output buffer too small for segment

## APPENDIX F

### BIBLIOGRAPHY

#### 1. DATA BASES

Aschim, Frode, "Data Base Networks - An Overview," Management Informatics, Vol. 3, No., 1976.

Bleier, Robert E., "Treating Hierarchical Data Structures in the SDC Time-Shared Data Management System (TDMS)," 1967 ACM National Meeting, pp. 41-49.

Booth, Grayce M., Honeywell Information Systems, Phoenix, Arizona, "The Use of Distributed Data Bases in Information Networks," First International Conference on Computer Communication: Impacts and Implications, October 24-26, 1972.

Boyce, R.F., Chamberlin, D.D., King III, W.F., and Hammer, M.M., "Specifying Queries as Relational Expressions: SQUARE," IBM Technical Report RJ 1291, October 1973.

Chamberlin, D.D. and Boyce, R.F., "SEQUEL: A Structured English Query Language," Proc. 1974 ACM SIGFIDET Workshop, Ann Arbor, Michigan, (April 1974), pp. 249-264.

Chandra A.M., "Some Considerations in the Design of Homogeneous Distribution Data Bases," IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

CODASYL Development Committee, "An Information Algebra," Communications of the ACM, Vol. 4, (April 1962), pp. 190-204.

CODASYL Data Base Conversion Task Group, Final Report, Information Processing, Vol. 2, No. 1, January 1977.

CODASYL Systems Committee, "Feature Analysis of Generalized Data Base Management Systems," New York, May 1971.

CODASYL Data Base Task Group Report, ACM April 1971.

Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol. 13, No. 6, (June 1970), pp. 377-387.

Codd, E.F., "Seven Steps to Rendezvous with the Casual User," IFIT TC-2 Working Conference on Data Base Management Systems, Cargese, Corsica, 1-5 April 1974.

Computing Surveys, ACM, Vol. 8, No. 1, March 1976.

Date, C.J. "An Architecture for High-Level Language Data Base Extensions," 1976 SIGMOD Conference.

Date, C.J., and Hopewell, P., "File Definition and Logical Data Independence," Proceedings of the 1971 ACM SIGFIDET Workshop.

Defense Intelligence Agency Regulation 65, Intelligence Information Systems, Defense Intelligence Data Standards System (IDSS).

Earley, Jay "Toward an Understanding of Data Structures," Comm. of the ACM Vol. 14, No. 10. October 1971.

Fredericksen, D.H., "Describing Data in a General Purpose Computer Network," IBM Research Report RC 4122, November 1972.

Fry, J.P. "Distributed Data Bases: A Summary of Research," The University of Michigan, Data Translation Project Working Paper, DE 801, August 1975.

Fry, J.P., Smith, D.P., and Taylor, R.W., "An Approach to Stored Data Definition and Translation," Proc. ACM SIGFIDET Workshop on Data Definition and Access, Denver, Colo., (1972), pp. 13-55.

Fry, J.P., "Stored Data Definition and Translation Approach to the Data Portability Problem," Data Translation Project Report, University of Michigan, Ann Arbor, Mich., February 1974.

Ghosh, S.P., and Senko, M.E., "String Path Search Procedures for Data Base Systems," IBM Journal of Research and Development, Vol. 18, No. 5, (September 1974), pp. 408-422.

Jervis, B., Parker, J., "An Approach for a Working Relational Data System," Department of Computer Science, University of British Columbia, Vancouver, Canada.

Logicon, Inc., "ADAPT I Uniform Data Language (UDL) A Preliminary Specification," (July 1976) San Diego, California.

Logicon, Inc., Final Report - Study of the Multi-Language Problem in COINS: Vol. 1. - Recommended Solutions, Vol. 2. - COINS DBMS Feature Analysis, Vol. 3. - Functional Specification. (May 1975) San Diego, California.

Lum, V.Y., Shu, N.C., Housel, B.C., "Data Translation, Part I: A General Methodology for Data Conversion and Restructuring," IBM (San Jose), JF 1525 (#23112), July 1975.



- Marcus, Richard S., "A Translating Computer Interface for a Network of Heterogeneous Interactive Information Retrieval Systems," MIT, Electronic Systems Laboratory, 1972.
- Merten, A.G., and Fry, J.P., "A Data Description Language Approach to File Translation," ACM Proc. SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Michigan, (1974), pp. 191-205.
- Plagman, G.K., and Altshuler, G.P., "A Data Dictionary/Directory System Within the Context of an Integrated Corporate Data Base," AFIPS, 1972, FJCC.
- Rosenthal, R., "A Review of Network Access Techniques with a Case Study: The Network Access Machine," National Bureau of Standards Technical Note 917, July 1976.
- Schneider, L.S., "A Relational View of the DIAM," Proceedings 1976 ACM SIGMOD International Conference on Management of Data, Washington, D.C., (June 1976), pp. 75-90.
- Senko, M.E., Altman, E.B., Astrahan, M.M., and Fehder, P.L., "Data Structures and Accessing in Data-Base Systems," IBM Systems Journal, Vol. 12, No. 1, (1973), pp. 30-93.
- Shoshani, Ari, "Data Sharing in Computer Networks," 1972 Wescon Tech. Papers, System Development Corp.
- Shoshani, Ari, and Spiegler, I., "The Integration of Data Management Systems on a Computer Network," American Institute of Aeronautics and Astronautics, Computer Network Systems Conference, Huntsville, Alabama, April 1973.
- Shoshani, Ari, and Brandon, K., "The Implementation of a Logical Data Base Converter," October 1975, System Development Corporation, TM-5590/000/00.
- Shu, N.C., Housel, B.C., and Lum, V.Y., "Data Translation, Part III. CONVERT: A High Level Translation Definition Language for Data Conversion." IBM Res. Rep. RJ 1515, February 1975.
- Smith, Diane P., "An Approach to Data Description and Conversion," RhD dissertation, University of Pennsylvania, Philadelphia, 1971.
- Taylor, R.W. "Generalized Data Base Management System Data Structures and their Mapping to Physical Storage," PhD dissertation, University of Michigan, Ann Arbor, 1971, University Microfilms 72-15014.



2. INCO PUBLICATIONS

Development Planning Factors for a Transparent Integrated Intelligence Network, INCO, INC., 1974.

Russek, Marianne, Distributed Data Base Information Systems Technology, INCO, INC., McLean, Virginia, 1974.

The Transparent Intelligence Language Facility, Final Technical Report, INCO, INC., McLean, Virginia, June 1976.

Transparent Integrated Intelligence Network Query Intermediate Processor, INCO, INC., McLean, Virginia, January 1977, RADC-TR-77-39, AD-A037 947/9WC.

TIIN Terminal Access Procedures Final Technical Report, INCO, INC., McLean, Virginia, January 1978.

3. LANGUAGE MANUALS

DIAOLS ADP User's Guide for On-Line and Remote/Local Batch Operations, DIA, Washington, D.C., 1974.

On-Line System Support Center (OSSC) User's Guide to DIAOLS/COINS, (to be published).

SEAWATCH External User's Manual, NOSIC, 1974.

TIPS Interrogation Language (TILE) Training Manual for NSA TIPS and COINS Users, NSA, 1972.

TOSS Information Management System (TIMS) On-Line User's Manual, July 1975, INCO, INC., McLean, Virginia.

## **MISSION of Rome Air Development Center**

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C<sup>3</sup>) activities, and in the C<sup>3</sup> areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

